On the Origins of Bisimulation and Coinduction

Davide Sangiorgi
University of Bologna, Italy
http://www.cs.unibo.it/~sangio/

June 24, 2008

Abstract

The origins of bisimulation and bisimilarity are examined, in the three fields where they have been independently discovered: Computer Science, Philosophical Logic (precisely, Modal Logic), Set Theory.

Bisimulation and bisimilarity are coinductive notions, and as such are intimately related to fixed points, in particular greatest fixed points. Therefore also the appearance of coinduction and fixed points is discussed, though in this case only within Computer Science. The paper ends with some historical remarks on the main fixed-point theorems (such as Knaster-Tarski) that underpin the fixed-point theory presented.

Contents

1 Introduction

Bisimulation and coinduction are generally considered as one of the most important contributions of Concurrency Theory to Computer Science. In concurrency, the bisimulation equality, called *bisimilarity*, is the most studied form of behavioural equality for processes, and is widely used, for a number of reasons, notably the following ones.

- Bisimilarity is accepted as *the finest behavioural equivalence* one would like to impose on processes.
- The *bisimulation proof method* is exploited *to prove equalities* among processes. This occurs even when bisimilarity is not the behavioural equivalence chosen for the processes. For instance, one may be interested in trace equivalence and yet use the bisimulation proof method since bisimilarity implies trace equivalence.
- The efficiency of the algorithms for bisimilarity checking and the compositionality properties of bisimilarity are exploited to *minimise* the state-space of processes.
- Bisimilarity, and variants of it such as similarity, are used *to abstract* from certain details of the systems of interest. For instance, we may want to prove behavioural properties of a server that do not depend on the data that the server manipulates. Further, abstracting from the data may turn an infinite-state server into a finite one.

Bisimulation has also spurred the study of coinduction; indeed bisimilarity is an example of a coinductive definition, and the bisimulation proof method an instance of the coinduction proof method.

Bisimulation and, more generally, coinduction are employed today in a number of areas of Computer Science: functional languages, object-oriented languages, types, data types, domains, databases, compiler optimisations, program

analysis, verification tools, etc.. For instance, in Type Theory bisimulation and coinductive techniques have been proposed: to prove the soundness of type systems [MT91]; to define the meaning of equality between (recursive) types and then to axiomatise and prove such equalities [AC93, BH97]; to define coinductive types and manipulate infinite proofs in theorem provers [Coq93, Gim96].

Also, the development of Final Semantics [Acz88, Acz93, RT92, JR96], an area of Mathematics based on coalgebras and category theory and used in the semantics of programming languages, has been largely motivated by the interest in bisimulation. Final Semantics also gives us a rich and deep perspective on the meaning of coinduction and its duality with induction.

In this paper, we look at the origins of bisimulation (and bisimilarity). We show that bisimulation has been discovered not only in Computer Science, but also—and roughly at the same time—in other fields: Philosophical Logic (more precisely, Modal Logic), and Set Theory. In each field, we discuss the main steps that led to the discovery, and introduce the people who made these steps possible.

In Computer Science, and in Philosophical Logic, and in Set Theory, bisimulation has been derived through refinements of notions of morphism between algebraic structures. Roughly, morphisms are maps that are "structure-preserving". The notion is therefore fundamental in all mathematical theories in which the objects of study have some kind of structure, or algebra. The most basic forms of morphism are the *homomorphisms*. These essentially give us a way of embedding a structure (the source) into another one (the target), so that all the relations in the source are present in the target. The converse however, need not be true; for this, stronger notions of morphism are needed. One such notion is *isomorphism*, which is however extremely strong—isomorphic structures must be essentially the same, i.e., "algebraically identical". It is a quest for notions in between homomorphism and isomorphism that led to the discovery of bisimulation.

The kind of structures studied in Computer Science, in Philosophical Logic, and in Set Theory were forms of rooted directed graphs. On such graphs bisimulation is coarser than graph isomorphism because, intuitively, bisimulation allows us to observe a graph only through the movements that are possible along its edges. By contrast, with isomorphisms the identity of the nodes is observable too. For instance, isomorphic graphs have the same number of nodes, which need not be the case for bisimilar graphs (bisimilarity on two graphs indicates that their roots are related in a bisimulation).

The independent discovery of bisimulation in three different fields suggests that only limited exchanges and contacts among researchers existed at the time. The common concept of bisimulation has somehow helped to improve this situation. An example of this are the advances in Set Theory and Computer Science derived from Aczel's work.

Bisimilarity and the bisimulation proof method represent examples of a coinductive definition and the coinduction proof method, and as such are intimately related to fixed points, in particular greatest fixed points. We therefore also discuss the introduction of fixed points, and of coinduction. In this case, however,

with a more limited breadth: we only consider Computer Science—fixed points have a much longer history in Mathematics—and we simply discuss the main papers in the introduction of coinduction and fixed-point theory in the field. We conclude with some historical remarks on the main fixed-point theorems that underpin all the fixed-point theory presented.

In each section of the paper, we focus on the origins of the concept dealt with in that section, and do not attempt to follow the subsequent developments. The style of presentation is generally fairly informal, but—we hope—technical enough to make the various contributions clear, so that the reader can appreciate them.

Structure of the paper In Section 2 we recall basic notions and results, whose history is then examined in the paper: bisimulation, bisimilarity, inductive characterisations of bisimilarity, coinduction, fixed-point theorems. In Section 3 to 5 we examine the origins of bisimulation and bisimilarity in Modal Logic, Computer Science, and Set Theory. In Section 6 we report on the introduction of coinduction and fixed points in Computer Science. Finally, in Section 7, we discuss the fixed-point theorems.

Acknowledgements I am very grateful to the following people who helped me to find relevant papers and materials or helped me in tracing back bits of history: P. Aczel, G. Boudol, J. van Benthem, E. Clarke, Y. Deng, R. Hinnion, F. Honsell, A. Mazurkiewicz, Y. N. Moschovakis, L. Moss, R. Milner, U. Montanari, W.P. de Roever, W. Thomas, M. Tofte. Thanks also to L. Aceto and to the referees for many comments and suggestions on an earlier draft.

2 Background

In this section we recall some basic notations, definitions, and results, including the definition of bisimulation, that are important in the remainder of the paper.

2.1 Bisimulation

We present bisimulation on *Labelled Transition Systems* (LTSs) because these are the most common structures on which bisimulation has been studied. LTSs are essentially labelled directed graphs. Bisimulation can be defined on variant structures, such as relational structures (i.e., unlabeled directed graphs) or Kripke structures, in a similar way; we will meet some of the variants in the following sections.

We let $\mathcal{R}$ range over relations on sets, i.e., if $\wp$ denotes the powerset construct, then a relation $\mathcal{R}$ on a set W is an element of $\wp(W \times W)$. The composition of relations $\mathcal{R}_1$ and $\mathcal{R}_2$ is $\mathcal{R}_1\mathcal{R}_2$ (i.e., $(s, s') \in \mathcal{R}_1\mathcal{R}_2$ holds if for some s'', both $(s, s'') \in \mathcal{R}_1$ and $(s'', s') \in \mathcal{R}_2$ hold). We often use the infix notation for relations; hence $s\ \mathcal{R}\ t$ means $(s, t) \in \mathcal{R}$.

Definition 2.1 (Relational structures) *A* relational structure *is a pair* $(W, \mathcal{T})$ *where* W *is a non-empty set called the* domain *of the structure, and* $\mathcal{T}$ *is a relation on* W*.*

We can think of LTSs as a kind of multi-relational structures.

Definition 2.2 (Labelled Transition Systems) *A* Labelled Transition System *is a triple* $(W, Act, \{\stackrel{a}{\longrightarrow} : a \in Act\})$ *with domain* W *as above, set of* labels *Act, and for each label* a*, a relation* $\stackrel{a}{\longrightarrow}$ *on* W *called the* transition relation*.*

In the two definitions above, the elements of W will be called *states* or *points*, sometimes even *processes* as this is the usual terminology in concurrency. We use s, t to range over such elements, and μ to range over the labels in *Act*. Following the infix notation for relations, we write $s \stackrel{\mu}{\longrightarrow} t$ when $(s,t) \in \stackrel{\mu}{\longrightarrow}$; in this case we call t a *μ-derivative of s*, or sometimes simply a *derivative of s*.

Definition 2.3 (Bisimulation) *A binary relation* $\mathcal{R}$ *on the states of an LTS is a* bisimulation *if whenever* $s_1 \mathcal{R} s_2$*:*

- *for all* s_1' *with* $s_1 \stackrel{\mu}{\longrightarrow} s_1'$*, there is* s_2' *such that* $s_2 \stackrel{\mu}{\longrightarrow} s_2'$ *and* $s_1' \mathcal{R} s_2'$*;*
- *the converse, on the transitions emanating from* s_2*.*

Bisimilarity*, written* $\sim$*, is the union of all bisimulations; thus* $s \sim t$ *holds if there is a bisimulation* $\mathcal{R}$ *with* $s \mathcal{R} t$*.*

The definition of bisimilarity has a strong impredicative flavor, for bisimilarity itself is a bisimulation and is therefore part of the union from which it is defined. Also, the definition immediately suggests a proof technique: to demonstrate that s_1 and s_2 are bisimilar, find a bisimulation relation containing the pair (s_1, s_2). This is the *bisimulation proof method*.

We will not discuss here the effectiveness of this proof method; the interested reader may consult concurrency textbooks in which bisimilarity is taken as the main behavioural equivalence for processes, such as [Mil89, SW01]. We wish however to point out two features of the definition of bisimulation that make its proof method practically interesting:

- the *locality* of the checks on the states;
- the lack of a *hierarchy* on the pairs of the bisimulation.

The checks are local because we only look at the immediate transitions that emanate from the states. An example of a behavioural equality that is non-local is *trace equivalence* (two states are trace equivalent if they can perform the same sequences of transitions). It is non-local because computing a sequence of transitions starting from a state s may require examining other states, different from s.

There is no hierarchy on the pairs of a bisimulation in that no temporal order on the checks is required: all pairs are on a par. As a consequence, bisimilarity

can be effectively used to reason about infinite or circular objects. This is in sharp contrast with inductive techniques, that require a hierarchy, and that therefore are best suited for reasoning about finite objects. For instance, here is a definition of equality that is local but inherently inductive:

> $s_1 = s_2$ if:
> for all s_1' with $s_1 \xrightarrow{\mu} s_1'$, there is s_2' such that $s_2 \xrightarrow{\mu} s_2'$ and $s_1' = s_2'$;
> the converse, on the transitions from s_2.

This definition requires a hierarchy, as the checks on the pair (s_1, s_2) must follow those on derivative pairs such as (s_1', s_2'). Hence the definition is ill-founded if the state space of the derivatives reachable from (s_1, s_2) is infinite or includes loops.

In the paper we also sometimes mention *simulations*, which are "half bisimulations".

Definition 2.4 (Simulation) *A binary relation $\mathcal{R}$ on the states of an LTS is a* simulation *if $s_1 \mathcal{R} s_2$ implies that for all s_1' with $s_1 \xrightarrow{\mu} s_1'$ there is s_2' such that $s_2 \xrightarrow{\mu} s_2'$ and $s_1' \mathcal{R} s_2'$.* Similarity *is the union of all simulations.*

We have presented the standard definitions of bisimulation and bisimilarity. A number of variants have been proposed and studied. For instance, on LTSs in which labels have a structure, which may be useful when processes may exchange values in communications; or on LTSs equipped with a special action to represent movements internal to processes, in which case one may wish to abstract from such action in the bisimulation game yielding the so-called *weak bisimulations* and *weak bisimilarity*. Examples of these kinds may be found, e.g., in [Mil89, SW01, AILS07, SKS07]. Also, we do not discuss in this paper enhancements of the bisimulation proof method, intended to relieve the amount of work needed to prove bisimilarity results, such as *bisimulation up-to* techniques; see, e.g., [Mil89, San98, Pou07].

2.2 Approximants of bisimilarity

We can approximate bisimilarity using the following inductively-defined relations and their meet. Similar constructions can be given for similarity.

Definition 2.5 (Stratification of bisimilarity) *Let W be the states of an LTS. We set:*

- $\sim_0 \stackrel{\text{def}}{=} W \times W$
- *$s \sim_{n+1} t$, for $n \geq 0$, if*
  1. *for all s' with $s \xrightarrow{\mu} s'$, there is t' such that $t \xrightarrow{\mu} t'$ and $s' \sim_n t'$;*
  2. *the converse, i.e., whenever for all t' with $t \xrightarrow{\mu} t'$, there is s' such that $s \xrightarrow{\mu} s'$ and $s' \sim_n t'$.*

- $\sim_\omega \stackrel{\text{def}}{=} \bigcap_{n\geq 0} \sim_n$

In general, $\sim_\omega$ does not coincide with $\sim$, as the following example shows.

Example 2.6 *Suppose $a \in Act$, and let a^0 be a state with no transitions, a^ω a state whose only transition is*

$$a^\omega \stackrel{a}{\longrightarrow} a^\omega \ ,$$

and a^n, for $n \geq 1$, states with only transitions

$$a^n \stackrel{a}{\longrightarrow} a^{n-1} \ .$$

Also, let s, t be states with transitions

$$s \stackrel{a}{\longrightarrow} a^n \qquad \text{for all } n \geq 0$$

and

$$t \stackrel{a}{\longrightarrow} a^n \qquad \text{for all } n \geq 0$$
$$t \stackrel{a}{\longrightarrow} a^\omega$$

It is easy to prove, by induction on n, that, for all n, $s \sim_n t$, hence also $s \sim_\omega t$. However, it holds that $s \not\sim t$: the transition $t \stackrel{a}{\longrightarrow} a^\omega$ can only be matched by s with one of the transitions $s \stackrel{a}{\longrightarrow} a^n$. But, for all n, we have $a^\omega \not\sim a^n$, as only from the former state $n+1$ transitions are possible.

In order to reach $\sim$, in general we need to replace the ω-iteration that defines $\sim_\omega$ with a transfinite iteration, using the ordinal numbers. However, the situation changes if the LTS is *image-finite* (or *finite-branching*), meaning that for all s the set $\{s' \ : \ s \stackrel{\mu}{\longrightarrow} s', \text{ for some } \mu\}$ is finite. (In Example 2.6, the LTS is not image-finite.) In this case, the natural numbers are sufficient: Indeed we have:

Theorem 2.7 *On image-finite LTSs, relations $\sim$ and $\sim_\omega$ coincide.*

2.3 Coinduction

Intuitively, a set A is defined *coinductively* if it is the *greatest* solution of an inequation of a certain form; then the *coinduction proof principle* just says that any set that is a solution of the same inequation *is contained* in A. Dually, a set A is defined *inductively* if it is the *least* solution of an inequation of a certain form, and the *induction principle* then says that any other set that is a solution to the same equation *contains* A. Familiar inductive definitions and proofs can be formalised in this way. To see how bisimulation and its proof method fit the coinductive schema, let $(W, Act, \{\stackrel{a}{\longrightarrow} : \ a \in Act\})$ be an LTS, and consider the function $F_\sim : \wp(W \times W) \to \wp(W \times W)$ so defined:

$F_\sim(\mathcal{R})$ is the set of all pairs (s, t) such that:

1. for all s' with $s \stackrel{\mu}{\longrightarrow} s'$, there is t' such that $t \stackrel{\mu}{\longrightarrow} t'$ and $s' \ \mathcal{R} \ t'$.

2. for all t' with $t \xrightarrow{\mu} t'$, there is s' such that $s \xrightarrow{\mu} s'$ and $s' \mathcal{R} t'$.

We call $F_\sim$ the *functional associated to bisimulation*, for we have:

Proposition 2.8 *1. $\sim$ is the greatest fixed point of $F_\sim$;*

2. $\sim$ is the largest relation $\mathcal{R}$ such that $\mathcal{R} \subseteq F_\sim(\mathcal{R})$; thus $\mathcal{R} \subseteq \sim$ for all $\mathcal{R}$ with $\mathcal{R} \subseteq F_\sim(\mathcal{R})$.

Proposition 2.8 is a simple application of fixed-point theory, in particular the Knaster-Tarski Theorem, that we discuss below. We recall that a *complete lattice* is a partially ordered set with all joins (i.e., all its subsets have a supremum, also called least upper bound); this implies that there are also all meets (i.e., all subsets have an infimum, also called greatest lower bound). Using $\leq$ to indicate the partial order, a point x in the lattice is a *post-fixed point* of an endofunction F on the lattice if $x \leq F(x)$; it is a *pre-fixed point* if $F(x) \leq x$.

Theorem 2.9 (Knaster-Tarski) *On a complete lattice, a monotone endofunction has a complete lattice of fixed points. In particular the greatest fixed point of the function is the join of all its post-fixed points, and the least fixed point is the meet of all its pre-fixed points.*

We deduce from the theorem that:

- a monotone endofunction on a complete lattice has a greatest fixed point;
- for an endofunction F on a complete lattice the following rule is sound:

$$\frac{F \text{ monotone} \qquad x \leq F(x)}{x \leq \mathrm{gfp}\,(F)} \tag{1}$$

  where $\mathrm{gfp}\,(F)$ indicates the greatest fixed point of F.

The existence of the greatest fixed point justifies coinductive definitions, while rule (1) expresses the coinduction proof principle, à la Knaster-Tarski.

Proposition 2.8 is a consequence of the Knaster-Tarski theorem because the functional associated to bisimulation gives us precisely the clauses of a bisimulation, and is monotone on the complete lattice of the relations on $W \times W$, in which the join is given by relational union, the meet by relational intersection, and the partial order by relation inclusion:

Lemma 2.10 • *$\mathcal{R}$ is a bisimulation iff $\mathcal{R} \subseteq F_\sim(\mathcal{R})$;*

• *$F_\sim$ is monotone (that is, if $\mathcal{R} \subseteq \mathcal{S}$ then also $F_\sim(\mathcal{R}) \subseteq F_\sim(\mathcal{S})$).*

For such functional $F_\sim$, (1) asserts that any bisimulation only relates pairs of bisimilar states. Example 2.6 shows that $\sim_\omega$ is not a fixed point for it.

Also Theorem 2.7, about approximating bisimilarity using the natural numbers, can be seen as an application of fixed-point theory, in which one uses the extra hypothesis of cocontinuity of the functional. Let $\bigcap$ denote the meet operation of the complete lattice; then an endofunction on such a lattice is *cocontinuous* if for all sequences $\alpha_0, \alpha_1 \ldots$ of decreasing points in the lattice (i.e., $\alpha_i \geq \alpha_{i+1}$, for $i \geq 0$) we have $F(\bigcap_i \alpha_i) = \bigcap_i F(\alpha_i)$.

Theorem 2.11 *For a cocontinuous endofunction F on a complete lattice we have:*

$$\mathrm{gfp}\,(F) = \bigcap_{n\geq 0} F^n(\top)$$

where $\top$ is the top element of the lattice, and $F^n(\top)$ indicates the n-th iteration of F on $\top$:

$$\begin{array}{rcl} F^0(\top) & \stackrel{\mathrm{def}}{=} & \top \\ F^{n+1}(\top) & \stackrel{\mathrm{def}}{=} & F(F^n(\top)) \end{array}$$

The cocontinuity of the functional associated to bisimilarity is guaranteed by the image-finite property of the LTS, and thus Theorem 2.7 becomes an instance of Theorem 2.11.

Without cocontinuity, to reach the greatest fixed point using inductively-defined relations we need to iterate over the transfinite ordinals, as the following theorem shows.

Theorem 2.12 *If F is a monotone endofunction on a complete lattice, then there is an ordinal α of cardinality less than or equal to that of the lattice such that for $\beta \geq \alpha$ the greatest fixed point of F is $F^\beta(\top)$ where $\top$ is the top element of the lattice and $F^\lambda(\top)$, where λ is an ordinal, is so defined:*

$$\begin{array}{rcll} F^0(\top) & \stackrel{\mathrm{def}}{=} & \top & \\ F^\lambda(\top) & \stackrel{\mathrm{def}}{=} & F(\bigcap_{\beta<\lambda} F^\beta(\top)) & \text{for } \lambda > 0 \end{array}$$

As the ordinals are linearly ordered, and each ordinal is either the successor of another ordinal or the least upper bound of all its predecessors, the above definition can also be given thus:

$$\begin{array}{rcll} F^0(\top) & \stackrel{\mathrm{def}}{=} & \top & \\ F^{\lambda+1}(\top) & \stackrel{\mathrm{def}}{=} & F(F^\lambda(\top)) & \text{for a successor ordinal} \\ F^\lambda(\top) & \stackrel{\mathrm{def}}{=} & F(\bigcap_{\beta<\lambda} F^\beta(\top)) & \text{for a limit ordinal} \end{array}$$

Theorem 2.12 tells us that at some ordinal α the function reaches its greatest fixed point. On ordinals larger than α, of course, the function remains on such fixed point. Therefore essentially the theorem tells us that $F^\lambda(\top)$ returns the greatest fixed point of F for all sufficiently large ordinals λ. In case F is cocontinuous, Theorem 2.11 assures us that we can take α to be the first limit ordinal, ω (not counting 0 as a limit ordinal). The property dual to cocontinuity, on increasing sequences, least fixed points and joins, is called *continuity*.[1]

Theorems 2.11 and 2.12 give us constructive proofs of the existence of greatest-fixed points. The constructions are indeed at the heart of the algorithms used today for bisimilarity checking.

[1] In some textbooks, cocontinuitiy is called *lower-continuity*, the dual property *upper-continuity*.

Complete lattices are "dualisable" structures: we can reverse the partial order and get another complete lattice. Thus the definitions and results above about joins, post-fixed points, greatest fixed points, cocontinuity have a dual in terms of meets, pre-fixed points, least fixed points, and continuity. As the results we gave justify coinductive definitions and the coinductive proof method, so the dual theorems can be used to justify familiar inductive definitions and inductive proofs for sets. However, to go deeper into coinduction and to fully appreciate the duality of concepts found in the theory of induction and in that of coinduction, it can be useful to go beyond the simple fixed-point theory above, and use the theory of algebras and coalgebras [Acz88, Acz93, JR96, TP97]. The fixed-point approach outlined in this section will be however sufficient for this paper.

Another well-known example of application of coinduction is in definition and proofs involving *divergence*. Divergence represents an infinite computation and can be elegantly defined coinductively; then the coinduction proof method can be used to prove that specific computations diverge.

3 Bisimulation in Modal Logic

3.1 Modal logics

Philosophical Logic studies and applies logical techniques to problems of interest to philosophers, somewhat similarly to what Mathematical Logic does for problems that interest mathematicians. Of course, the problems do not only concern philosophers or mathematicians; for instance nowadays both philosophical and mathematical logics have deep and important connections with Computer Science.

Strictly speaking, in Philosophical Logic a modal logic is any logic that uses *modalities*. A modality is an operator used to qualify the truth of a statement, that is, it creates a new statement that makes an assertion about the truth of the original statement. Originally, modal logic was just the logic of *necessity* and *possibility*, to express assertions of the form "it is possible that", or "it is necessary that". Nowadays it has a broader connotation, as the term is used also to refer to other logics: temporal logics (also called tense logics), that talk about future and past (to express assertions such as "it always will be that", or "it was the case that"); epistemic logics, that talk about the certainty of sentences (to express assertions such as "it is certainly true that", or "it may be true that [given the available knowledge]"); deontic logics, that talk about obligation and morality (to express assertions such as "it is obligatory that", "it is permitted that", etc.); and so forth.

For the discussion below we use a simple modal language, defined by means of the usual operators of propositional logic, a set of *proposition letters* $\{p_i\}_{i\in I}$, and a unary modal operator $\Diamond$:

$$\phi \stackrel{\text{def}}{=} p \;\Big|\; \neg\phi \;\Big|\; \phi_1 \wedge \phi_2 \;\Big|\; \Diamond\phi \;\Big|\; \bot$$

where p is a proposition letter. The arguably simplest form of semantics for modal languages is the *Kripke semantics*. Here, models are directed graphs whose nodes are labelled with sets of proposition letters. Formally, a *model* is a triple $(W, \mathcal{T}, G)$ where $(W, \mathcal{T})$ is a relational structure (Definition 2.1; relational structures are usually called *frames* in Modal Logic); and G is a function that assigns to each point in W a set of proposition letters, to be thought of as the atomic formulas that hold at that point. As usual for relations, we write $s \mathcal{T} t$ if $(s,t) \in \mathcal{T}$; in this case we say that there is a *transition* between s and t. The notion of satisfaction for a formula at a point t of a model M is defined inductively thus:

M,t	$\models$	p	if $p \in G(t)$
M,t	$\models$	$\bot$	never holds
M,t	$\models$	$\phi_1 \wedge \phi_2$	if both $M,t \models \phi_1$ and $M,t \models \phi_2$
M,t	$\models$	$\neg\phi$	if not $M,t \models \phi$
M,t	$\models$	$\Diamond\phi$	if for some s such that $t \mathcal{T} s$ we have $M,s \models \phi$

Thus $\Diamond\phi$ holds at t if ϕ holds in at least one of the successors of t; and p holds at t if p is among the letters assigned to t; the interpretation of the other operators is the standard one of propositional logic. While models are used to investigate questions of satisfaction of formulas, relational structures are used for questions of validity.

To keep things simple we have assumed a single transition relation $\mathcal{T}$ and a single modality $\Diamond$, but in fact one could have a set of modalities, say $\langle a \rangle$, where a is taken from some special alphabet, and for each modality $\langle a \rangle$ a corresponding transition relation $\mathcal{T}_a$; thus Labelled Transition Systems become special cases of models, namely models with a separate modality and transition relation for each different form of action, and without proposition letters. In the examples we will give, when we do not mention proposition letters it is intended that no proposition letters hold at the states under consideration. Further, we write $\mathcal{T}^M$ for the transition relation of a model M.

3.2 From homomorphism to p-morphism

Today, some of the most interesting results in the expressiveness of modal logics rely on the notion of bisimulation. Bisimulation is indeed discovered in Modal Logic when researchers begin to investigate seriously issues of expressiveness for the logics, in the 1970s. For this, important questions tackled are: When is the truth of a formula preserved when the model changes? Or, even better, under which model constructions are modal formulas invariant? Which properties of models can modal logics express? (When moving from a model M to another model N, preserving a property means that if the property holds in M then it holds also when one moves to N; the property being invariant means that also the converse is true, that is, the property holds in M iff it holds when one moves to N.)

To investigate such questions, it is natural to start from the most basic structure-preserving construction, that of *homomorphism*. A homomorphism

from a model M to a model N is a function F from the points of M to the points of N such that

- whenever a proposition letter holds at a point s of M then the same letter also holds at $F(s)$ in N;
- whenever there is a transition between two points s, s' in M then there is also a transition between $F(s)$ and $F(s')$ in N.

Thus, contrasting homomorphism with bisimulation, we note that

(i) homomorphism is a functional, rather than relational, concept;

(ii) in the definition of homomorphism there is no back condition; i.e., the reverse implication, from transitions in N to those in M, is missing.

Homomorphisms are too weak to respect the truth of modal formulas. That is, a homomorphism H from a model M to a model N does not guarantee that if a formula holds at a point t of M then the same formula also holds at $H(t)$ in N. For instance, consider a model M with just one point and no transitions, and a model N with two points and transitions between them. A homomorphism can send the point of M onto any of the points of N. The formula $\neg\Diamond\neg\bot$, however, which holds at points that have no transitions, will be true in M, and false in N.

The culprit for the failure of homomorphisms is the lack of a back condition. We can therefore hope to repair the invariance by adding some form of reverse implication. There are two natural ways of achieving this:

1. turning the "implies" of the definition of homomorphism into an "iff" (that is, a propositional letter holds at s in M iff it holds at $F(s)$ in N; and $s \mathcal{T}^M s'$ in M iff $F(s) \mathcal{T}^N F(s')$ in N);
2. explicitly adding a back condition (that is, if in N there is a transition $F(s) \mathcal{T}^N t$, for some point t, then in M there exists a point s' such that $s \mathcal{T}^M s'$ and $t = F(s')$.

Solution (1) is the requirement of *strong homomorphisms*. Solution (2) is first formalised by Krister Segerberg in his famous dissertation [Seg71], as the requirement of *p-morphisms*.

Segerberg starts the study of morphisms between models of modal logics that preserve the truth of formulas in [Seg68]. Initially, p-morphisms are called *pseudo-epimorphims* [Seg68], and are indeed surjective mappings. Later [Seg70, Seg71], the term is shortened to p-morphisms, and thereafter used to denote also non-surjective mappings. A notion similar to p-morphisms had also occurred earlier, in a work of Jongh and Troelstra [JT66], for certain surjective mappings on partial orders that were called *strongly isotone*. These were in fact essentially pseudo-epimorphisms on frames, rather than on models; they had been used to study relationships between partial orders and algebraic models of intuitionistic propositional logic. Sometimes, today, p-morphisms are called

bounded morphisms, after Goldbatt [Gol89]. The p-morphisms can be regarded as the natural notion of homomorphism in Kripke models; indeed other reasons make p-morphisms interesting for modal logics, for instance they are useful in the algebraic semantics of modal logics (e.g., when relating modal algebras).

With either of the additions in (1) or (2), the invariance property holds: modal formulas are invariant both under surjective strong homomorphisms and under p-morphisms. Thus, if H is a surjective strong homomorphism, or a p-morphism, from M to N, then for any point s in M and formula ϕ, we have $M, s \models \phi$ iff $N, H(s) \models \phi$. (The surjective condition is necessary for strong homomorphisms, but not for p-morphisms.)

As far as invariance is concerned, the surjective strong homomorphism condition is certainly a very strong requirement—we are not far from isomorphism, in fact (the only difference is injectivity of the function, but even when functions are not injective only states with essentially the "same" transitions can be collapsed, that is, mapped onto the same point). In contrast, p-morphisms are much more interesting. Still, they do not capture all situations of invariance. That is, there can be states s of a model M and t of a model N that satisfy exactly the same modal formulas and yet there is no p-morphisms that take s into t or vice versa.

3.3 Johan van Benthem

The next step is made by Johan van Benthem in his PhD thesis [Ben76] (the book [Ben83] is based on the thesis), who generalises the directional relationship between models in a p-morphism (the fact that a p-morphism is a function) to a symmetric one. This leads to the notion of bisimulation, which van Benthem calls *p-relation*. (Later [Ben84] he renames p-relations as *zigzag relations*.) On Kripke models, a p-relation between models M and N is a total relation $\mathcal{S}$ on the states of the models (the domain of $\mathcal{S}$ are the states of M and the codomain the states of N) such that whenever $v \mathcal{S} t$ then: a propositional letter holds at s iff it holds at t; for all s' with $s \mathcal{T}^M s'$ there is t' such that $t \mathcal{T}^N t'$ and $s' \mathcal{S} t'$; the converse of the previous condition, on the transitions from t.

To appreciate the difference between p-morphisms and p-relations, consider the models in Figure 1 (where the letters are used to name the states, they do not represent proposition letters—there are no proposition letters, in fact). There is no p-morphisms from M to N: the image of t must be either v or w; in any case, there is always a transition from u that s cannot match. We can however establish a p-relation on the models, relating s with u, and t with both v and w. (There is a p-morphism in the opposite direction, from N to M; but the example could be developed a bit so that there is no p-morphisms in either direction.)

Van Benthem defines p-relations while working on *Correspondence Theory*, precisely the relationship between modal and classical logics. Van Benthem's objective is to characterise the fragment of first-order logic that "corresponds" to modal logic—an important way of measuring expressiveness. He gives a sharp answer to the problem, via a theorem that is today called "van Benthem

Figure 1: On p-morphisms and p-relations

Characterisation Theorem". We explain van Benthem's result in some detail because both its assertion and its proof are interesting.

Van Benthem is not the first one to compare modal logic and first-order logic. There is a straightforward translation of modal logic into first-order logic, usually called the "standard translation" that had been known for a long time (since the 1960s, at least). That such a translation should exist is by no means surprising: Kripke models—the models of modal logic—can also be talked about using classical logic. With first-order logic, for instance, we simply need a relational symbol to match the relations on the points of the model and a set of predicates to match the proposition letters that hold at such points. Thus the standard translation maps modal formulas to the formulas of the language FO of first-order logic that has equality, a binary relation $\mathcal{U}$, and unary predicates $\{P_i \ : \ p_i \text{ is a proposition letter in the modal logic}\}$. The translation, $[\![\phi]\!]$, is defined as follows on the structure of ϕ:

$$
\begin{array}{rcll}
[\![p]\!]_x & \stackrel{\text{def}}{=} & Px & \\
[\![\bot]\!]_x & \stackrel{\text{def}}{=} & x \neq x & \\
[\![\neg\phi]\!]_x & \stackrel{\text{def}}{=} & \neg[\![\phi]\!]_x & \\
[\![\phi \vee \psi]\!]_x & \stackrel{\text{def}}{=} & [\![\phi]\!]_x \vee [\![\psi]\!]_x & \\
[\![\Diamond\phi]\!]_x & \stackrel{\text{def}}{=} & \exists y.(x \,\mathcal{U}\, y \wedge [\![\phi]\!]_y) & \text{with } x \neq y
\end{array}
$$

The free variable x in $[\![\phi]\!]_x$ is needed for evaluation of the formula: the evaluation of ϕ at a point t corresponds to the evaluation of $[\![\phi]\!]_x$ where t replaces x. Indeed in any Kripke model, ϕ holds at a point t iff $[\![\phi]\!]_x\{t/x\}$ holds. Note the translation of the modality $\Diamond$, which makes it explicit that a modal logic implicitly gives us a form of quantification, yet—in contrast with first-order logic—the quantification is achieved without the use of variables.

Although known for a long time, the importance of the translation emerges only in the 1970s, notably in works of Thomason [Tho72] and in those of van Benthem mentioned above. Van Benthem Characterisation Theorem relates modal logic to first-order logic via the translation and bisimulation. In the first-order language FO defined above, a formula A with one free variable, say x, is *invariant for bisimulations* if the evaluation of A at bisimilar points is always the same; that is, for all models M, N, and all states s in M and t in N, and all bisimulations $\mathcal{S}$ between M and N such that $s \,\mathcal{S}\, t$, we have $M \models A\{s/x\}$

iff $N \models A\{t/x\}$ (where $M \models A\{s/x\}$ means the formula A is satisfied in M, in the usual manner of first-order logics, when s is assigned to the free variable x; similarly for $N \models A\{t/x\}$). In nowadays's terminology, van Benthem's theorem says: a first-order formula A containing one free variable is equivalent to the standard translation of a modal formula iff A is invariant for bisimulations. That is, modal logic is the fragment of first-order logic whose formulas have one free variable and are invariant for bisimulation.

The standard translation therefore defines a *proper* fragment of first-order logic (not just syntactically, but also semantically). Indeed, with first-order logic one can define very intensional properties of states in the models. For instance the formula $\exists y.((x \neq y) \wedge (x \mathcal{U} y) \wedge (y \mathcal{U} x))$ says that there are 2 connected points, and the formula $x \mathcal{U} x$ says that there is at least one state with a transition onto itself. These properties can distinguish states that are bisimilar. For example the second formula distinguishes a state that has a transition onto itself from a bisimilar state that has non-terminating sequences of transitions but no transitions onto itself. Hence the formulas are not definable in modal logic since, as van Benthem's theorem shows, modal logic respects bisimulation. It is not so surprising that modal logic should correspond to a proper fragment of first-order logic. The strength of van Benthem's theorem is that it tells us precisely what this fragment is.

The actual assertion in van Benthem's original theorem is a bit different: in place of "invariant for bisimulations" van Benthem required that the formula is "invariant for p-relations and generated submodels". The reason for this is that p-relations are "total bisimulations", not arbitrary ones. Hence in general it is not possible to establish a p-relation between a model and submodels of it. Thus van Benthem, who heavily plays with models and submodels in the proof, needs a further restriction for the theorem to be true, namely the invariance for generated submodels. (Intuitively, M is a generated submodel of a model N if M is a subgraph of N that is transition closed, i.e., all transitions in N emanating from a point s that is also in M are also transitions of M.) Generated submodels represent an important construction on models of modal logics, and the property of invariance under generated submodels had already been used by other researchers before van Benthem (for instance, Feferman and Kreisel, in the 1960s).

The original proof of the theorem is also interesting. The implication from left to right is easy: one shows that modal formulas are invariant for bisimulation proceeding by induction on the depth of the formulas. The opposite implication is the difficult one. Here a key part of the proof is to show that a point s in a model M and a point t in a model N satisfy the same modal formulas if there are extensions M' and N' of the models M and N in which s and t are bisimilar. The extensions are obtained as the limits of appropriate elementary chains of models, starting from the original models. Further, the embedding of the original models into the limits of the chains preserves modal formulas. The reason why it is necessary to move from the original models M and N to the extended models M' and N' is that on arbitrary models two points may satisfy the same set of formulas without being bisimilar. This may occur if

the models are not image-finite. By contrast, the extended models M' and N' are "saturated", in the sense that they have "enough points". On such models, two points satisfy the same modal formulas iff they are bisimilar. As all image-finite models are saturated, van Benthem's construction also yields the familiar Hennessy-Milner Theorem for modal logics [HM85] (an earlier version is [HM80]): on image-finite models, two points are bisimilar iff they satisfy the same modal formulas. Saturated models need not be image-finite, however, thus van Benthem's construction is somewhat more general. Note that the need for saturation also would disappear if the logic allowed some infinitary constructions, for instance infinite conjunction.

Van Benthem does not isolate the above part of the proof as a separate lemma and this may have contributed to the result remaining unknown to researchers in other fields. (The Hennessy-Milner Theorem, for instance, appears some years later.)

For more details on van Benthem's proof, see [Ben76, Ben83]; in modern textbooks, such as [BRV01], the proof is sometimes presented in a different way, by directly appealing to the existence of saturated models; however elementary chains are employed to show the existence of such saturated models.

After van Benthem's theorem, bisimulation has been used extensively in Modal Logic, for instance, to analyze the expressive power of various dialects of modal logics, to understand which properties of models can be expressed in modal logics, to define operations on models that preserve the validity of modal formulas.

3.4 Discussion

In Philosophical Logic we see, historically, the first appearance of the notion of bisimulation. We do not find here, however, coinduction, at least not in an explicit way. Thus total relations between models that represent bisimulations are defined—the p-relations—but there is no explicit definition and use of bisimilarity. Similarly no links are made to fixed-point theory.

In retrospective, today we could say that bisimulation, as a means of characterising equivalence of modal properties "was already there" in the *Ehrenfeucht-Fraïssé games*. In the 1950s, Roland Fraïssé [Fra53] gave an algebraic formulation, as a weak form of isomorphism, of indistinguishability by formulas of first-order logic. Andrzej Ehrenfeucht [Ehr61] then extended the result and gave it a more intuitive game theoretic formulation, in what is now called the Ehrenfeucht-Fraïssé games. Such games are today widely used in Computer Science, notably in Logic, Finite Model Theory, but also in other areas such as Complexity Theory, following Immerman [Imm82]. It is clear that the restriction of the Ehrenfeucht-Fraïssé games to modal logic leads to game formulations of bisimulation. However, such a connection has been made explicit only after the discovery of bisimulation. See for instance Thomas [Tho93].

4 Bisimulation in Computer Science

4.1 Algebraic theory of automata

In Computer Science, the search for the origins of bisimulation takes us back to the algebraic theory of automata, well-established in the 1960s. A good reference is Ginzburg's book [Gin68]. Homomorphisms can be presented on different forms of automata. From the bisimulation perspective, the most interesting notions are formulated on *Mealy automata*. In these automata, there are no initial and final states; however, an output is produced whenever an input letter is consumed. Thus Mealy automata can be compared on the set of output strings produced. Formally, a Mealy automaton is a 5-tuple $(W, \Sigma, \Theta, \mathcal{T}, \mathcal{O})$ where

- W is the finite set of *states*;
- Σ is the finite set of *inputs*;
- Θ is a finite set of *outputs*;
- $\mathcal{T}$ is the *transition function*, that is a set of partial functions $\{\mathcal{T}_a \ : \ a \in \Sigma\}$ from W to W;
- $\mathcal{O}$ is the *output function*, that is, a set of partial functions $\{\mathcal{O}_a \ : \ a \in \Sigma\}$ from W to Θ.

The output string produced by a Mealy automaton is the *translation* of the input string with which the automaton was fed; of course the translation depends on the state on which the automaton is started. Since transition and output functions of a Mealy automaton are partial, not all input strings are consumed entirely.

Homomorphism is defined on Mealy automata following the standard notion in algebra, e.g., in group theory: a mapping that commutes with the operations defined on the objects of study. Below, if A is an automaton, then W^A is the set of states of A, and similarly for other symbols. As we deal with partial functions, it is convenient to view these as relations, and thereby use for them relational notations. Thus fg is the composition of the two function f and g where f is used first (that is, $(fg)(a) = g(f(a))$); for this, one requires that the codomain of f be included in the domain of g. Similarly, $f \subseteq g$ means that whenever f is defined then so is g, and they give the same result.

A *homomorphism* from the automaton A to the automaton B is a surjective function F from W^A to W^B such that for all $a \in \Sigma$:

1. $\mathcal{T}_a^A F \subseteq F \mathcal{T}_a^B$ (condition on the states); and
2. $\mathcal{O}_a^A \subseteq F \mathcal{O}_a^B$ (condition on the outputs).

(We assume here for simplicity that the input and output alphabets are the same, otherwise appropriate coercion functions would be needed.)

At the time (the 1960s), homomorphism and alike notions are all expressed in purely algebraic terms. Today we can make an operational reading of them,

which for us is more enlightening. Writing $s \xrightarrow[b]{a} t$ if the automaton, on state s and input a, produces the output b and evolves into the state t, and assuming for simplicity that $\mathcal{O}_a^A$ and $\mathcal{T}_a^A$ are undefined exactly on the same points, the two conditions above become:

- for all $s, s' \in W^A$, if $s \xrightarrow[b]{a} s'$ then also $F(s) \xrightarrow[b]{a} F(s')$.

Homomorphisms are used in that period to study a number of properties of automata. For instance, minimality of an automaton becomes the condition that the automaton has no proper homomorphic image. Homomorphisms are also used to compare automata. Mealy automata are compared using the notion of *covering* (written $\leq$): $A \leq B$ (read "automaton B covers automaton A") if B can do, statewise, at least all the translations that A does. That is, there is a total function ψ from the states of A to the states of B such that, for all states s of A, all translations performed by A when started in s can also be performed by B when started in $\psi(s)$. Note that B can however have states with a behaviour completely unrelated to that of any state of A; such states of B will not be the image of states of A. If both $A \leq B$ and $B \leq A$ hold, then the two automata are deemed *equivalent*.

Homomorphism implies covering, i.e., if there is a homomorphism from A to B then $A \leq B$. The converse result is (very much) false. The implication becomes stronger if one uses *weak homomorphisms*. These are obtained by relaxing the functional requirement of homomorphism into a relational one. Thus a weak homomorphism is a total relation $\mathcal{R}$ on $W^A \times W^B$ such that for all $a \in \Sigma$:

1. $\mathcal{R}^{-1}\mathcal{T}_a^A \subseteq \mathcal{T}_a^B\mathcal{R}^{-1}$ (condition on the states); and
2. $\mathcal{R}^{-1}\mathcal{O}_a^A \subseteq \mathcal{O}_a^B$ (condition on the outputs).

where relational composition, inverse, and inclusion are defined in the usual way for relations (and functions are taken as special forms of relations). In an operational interpretation as above, the conditions give:

- whenever $s \mathcal{R} t$ and $s \xrightarrow[b]{a} s'$ hold in A, then there is t' such that $t \xrightarrow[b]{a} t'$ holds in B and $s' \mathcal{R} t'$.

(On the correspondence between the algebraic and operational definitions, see also Remark 4.1 below.) Weak homomorphism reminds us of the notion of simulation for Labelled Transition Systems (LTSs). The former is however stronger, because the relation $\mathcal{R}$ is required to be *total*. (Also, in automata theory, the set of states and the sets of input and output symbols are required to be finite, but this difference is less relevant.)

Remark 4.1 To understand the relationship between weak homomorphisms and simulations, we can give an algebraic definition of simulation on LTSs, taking these to be triples $(W, \Sigma, \{\mathcal{T}_a \ : \ a \in \Sigma\})$ whose components have the same interpretation as for automata. A simulation between two LTSs A and B

Figure 2: On homomorphisms and weak homomorphisms

---

becomes a relation $\mathcal{R}$ on $W^A \times W^B$ such that, for all $a \in \Sigma$, condition (1) of weak homomorphism holds, i.e.

- $\mathcal{R}^{-1}\mathcal{T}_a^A \subseteq \mathcal{T}_a^B\mathcal{R}^{-1}$

This is precisely the notion of simulation defined operationally in Definition 2.4. Indeed, given a state $t \in W^B$ and a state $s' \in W^A$, we have $t\mathcal{R}^{-1}\mathcal{T}_a^A s'$ whenever there is $s \in W^A$ such that $s \xrightarrow{a} s'$. Then, requiring that the pair (t, s') is also in $\mathcal{T}_a^B\mathcal{R}^{-1}$ is the demand that there is t' such that $t \xrightarrow{a} t'$ and $s' \mathcal{R} t'$. □

As homomorphisms, so weak homomorphisms imply covering. The result for weak homomorphism is stronger as the homomorphisms are strictly included in the weak homomorphisms. An example of the strictness is given in Figure 2, where the states s_i belong to an automaton and the states t_i to another one, there are two input letters a and b, and for simplicity we ignore the automata outputs. We cannot establish a homomorphism from the automaton on the left to the automaton on the right, since a homomorphism must be surjective. Even leaving the surjective condition aside, a homomorphism cannot be established because the functional requirement prevents us from relating s_3 with both t_3 and t_4. By contrast, a weak homomorphism exists, relating s_1 with t_1, s_2 with t_2, and s_3 with both t_3 and t_4.

As weak homomorphisms are total relations, however, the converse is still false: covering does not imply weak homomorphism. Indeed I have not found, in the literature of that time, characterisations of covering, or equivalence, in terms of notions akin to homomorphism. Such characterisations would have taken us closer to the idea of bisimulation.

In conclusion: in the algebraic presentation of automata in the 1960s we find concepts that remind us of bisimulation, or better, simulation. However there are noticeable differences, as we have outlined above. But the most important difference is due to the fact that the objects are deterministic. To see how significant this is, consider the operational reading of weak homomorphism, namely "whenever $s \mathcal{R} t$ ... then there is t' such that....". As automata are deterministic, the existential in front of t' does not play a role. Thus the alternation of universal and existential quantifiers—a central aspect of the definitions of bisimulation and simulation—does not really show up on deterministic automata.

4.2 Robin Milner

Decisive progress towards bisimulation is made by Robin Milner in the 1970s. Milner transplants the idea of weak homomorphism into the study of the behaviour of programs in a series of papers in the early 1970s ([Mil70, Mil71a, Mil71b], with [Mil71b] being a synthesis of the previous two). He studies programs that are sequential, imperative, and that may not terminate. He works on the comparisons among such programs. The aim is to develop techniques for proving the correctness of programs, and for abstracting from irrelevant details so that it is clear when two programs are realisations of the same algorithm. In short, the objective is to understand when and why two programs can be considered "intensionally" equivalent.

To this end, Milner proposes—appropriately adapting it to his setting—the algebraic notion of weak homomorphism that we have described in Section 4.1. He renames weak homomorphism as *simulation*, a term that better conveys the idea of the application in mind. Although the definition of simulation is still algebraic, Milner now clearly spells out its operational meaning. But perhaps the most important contribution in his papers is the proof technique associated to simulation that he strongly advocates. This techniques amounts to exhibiting the set of pairs of related states, and then checking the simulation clauses on each pair. The strength of the technique is precisely the *locality* of the checks that have to be made, in the sense given to locality in Section 2.1. The technique is proposed to prove not only results of simulation, but also results of input/output correctness for programs, as a simulation between programs implies appropriate relationships on their inputs and outputs. Besides the algebraic theory of automata, other earlier works that have been influential for Milner are those on program correctness, notably Floyd [Flo67], Manna [Man69], and Landin [Lan69], who pioneers the algebraic approach to programs.

Milner's simulation is used by other authors, notably by Tony Hoare, in a paper [Hoa72] widely cited in the literature on programming languages, especially the object-oriented ones. Hoare uses simulation as a basis for a method of proving correctness of concrete representations of program data with respect to abstract versions of them.

Formally, however, Milner's simulation remains the same as weak homomorphism and as such it is not today's simulation. Programs for Milner are deterministic, with a total transition function, and these hypotheses are essential. Non-deterministic and concurrent programs or, more generally, programs whose computations are trees rather than sequences, are mentioned in the conclusions for future work. It is quite possible that if this challenge had been quickly taken up, then today's notion of simulation (or even bisimulation) would have been discovered much earlier.

Milner himself, later in the 1970s, does study concurrency very intensively, but under a very different perspective: he abandons the view of parallel programs as objects with an input/output behaviour akin to functions, in favor of the view of parallel programs as *interactive* objects. This leads Milner to develop a new theory of processes and a calculus—CCS—in which the notion of

behavioural equivalence between processes—*observational equivalence*—is fundamental. Milner however keeps, from his earlier works, the idea of "locality", that is, the interest in notions of equivalence in which outcomes are local to states, rather than global to programs like their traces.

The behavioural equivalence that Milner puts forward, and that is prominent in the first book on CCS [Mil80], is inductively defined. It is the stratification of bisimilarity, $\sim_\omega$, that we discuss in Section 2.2. Technically, in contrast with weak homomorphisms, $\sim_\omega$ has also the reverse implication (on the transitions of the second components of the pairs in the relation), and can be used on nondeterministic structures. The addition of a reverse implication was not obvious. For instance, a natural alternative would have been to maintain an asymmetric basic definition, possibly refine it, and then take the induced equivalence closure to obtain a symmetric relation (if needed). Indeed, among the main behavioural equivalences in concurrency—there are several of them, see [Gla93, Gla90])—bisimulation is the only one that is not naturally obtained as the equivalence-closure of a preorder (which incidentally also explains why giving denotational interpretations of bisimulation can be hard).

With Milner's advances, the notion of bisimulation is almost there: it remained to turn an inductive definition into a coinductive one. This will be David Park's contribution.

It is worth pointing out that, towards the end of the 1970s, homomorphisms-like notions appear in other attempts at establishing "simulations", or even "equivalences", between concurrent models—usually variants of Petri Nets. Good examples are John S. Gourlay, William C. Rounds, and Richard Statman [GRS79] and Kurt Jensen [Jen80], which develop previous work by Daniel Brand [Bra78] and Y. S. Kwong [Kwo77]. Gourlay, Rounds, and Statman's homomorphisms (called *contraction*) relate an abstract system with a more concrete realisation of it—in other words, a specification with an implementation. Jensen's proposal (called *simulation*), which is essentially the same as Kwong's *strict reduction* [Kwo77], is used to compare the expressiveness of different classes of Petri Nets. The homomorphisms in both papers are stronger than today's simulation or bisimulation; for instance they are functions rather than relations. Interestingly, in both cases there are forms of "reverse implications" on the correspondences between the transitions of related states. Thus these homomorphisms, but especially those in [GRS79], remind us of bisimulation, at least in the intuition behind it. In [GRS79] and [Jen80], as well as other similar works of that period, the homomorphisms are put forward because they represent conditions sufficient to preserve certain important properties (such as Church-Rosser and deadlock freedom). In contrast with Milner, little emphasis is given to the proof technique based on local checks that they bear upon. For instance the definitions of the homomorphisms impose correspondence on *sequences* of actions from related states.

4.3 David Park

In 1980[2] Milner returns to Edinburgh after a six-month appointment at Aarhus University, and completes his first book on CCS[3]. Towards the end of that year, David Park begins a sabbatical in Edinburgh, and stays at the top floor of Milner's house.

Park is one of the top experts in fixed-point theory at the time. He makes the final step in the discovery of bisimulation precisely guided by fixed-point theory. Park notices that the inductive notion of equivalence that Milner is using for his CCS processes is based on a monotone functional over a complete lattice. And by adapting an example by Milner, he sees that Milner's equivalence is not a fixed point for the functional, and that therefore the functional is not cocontinuous. He then defines bisimilarity as the greatest fixed point of the functional, and derives the bisimulation proof method from the theory of greatest fixed points. Further, Park knows that, to obtain the greatest fixed point of the functional in an inductive way, the ordinals and transfinite induction, rather then the naturals and standard induction, are needed (see also the discussion on Theorem 2.12 in Section 7). Milner immediately and enthusiastically adopts Park's proposal, and in the years to come makes it popular and the cornerstone of the theory of CCS [Mil89]. Here is the discovery of bisimulation, and the choice of the name for it, in Milner's own words:

> He [i.e., David Park] came down during breakfast one morning carrying my CCS book and said "there's something wrong!". So I prepared to defend myself. He pointed out the non coinductive way that I had set up observation equivalence, as the limit of a decreasing ω-chain of relations, which didn't quite reach the maximal fixed point.
>
> After about 10 minutes I realised he was right, and through that day I got excited about the coinductive proof technique.
>
> That was what David meant by "something's wrong". Not only had I missed the (fixed!) point—which I had realised—but also my proof technique (involving induction on the iteration of the functions) for establishing instances of the equivalences was clumsy. I immediately saw that he had liberated me from a misconception, and that the whole theory was going to look very much better by using maximal fixed points and (what I now recognise as) coinduction. [...]
>
> That same day we went for a walk in the hills around Edinburgh, and the express purpose was to agree what the pre-fixed points and the maximal fixed point should be called. We thought of a lot of words; David at one point liked "mimicry", which I vetoed. I think "bisimulation" was my suggestion; in any case, we both liked it, partly because we could use that word for the pre-fixed points and "bisimilarity" for the maximal fixed point itself. I think David demurred

[2],

[3]The first part of this section is based on personal communications with R. Milner.

> because there are five syllables; but we then thought that they were a lot easier to pronounce than the three syllables of "mimicry"!

Milner knew that $\sim_\omega$ is not invariant under transitions. Indeed he is not so much struck by the difference between $\sim_\omega$ and bisimilarity as behavioural equivalences, as the processes exhibiting such differences can be considered rather artificial. What excites him is the coinductive proof technique for bisimilarity. Both bisimilarity and $\sim_\omega$ are rooted in the idea of locality, but the coinductive method of bisimilarity further facilitates proofs.

In Computer Science, the standard reference for bisimulation and the bisimulation proof method is Park's paper "Concurrency on Automata and Infinite Sequences" [Par81a] (one of the most quoted papers in concurrency). While the reference to Park is justified, mentions of that particular paper can be questioned.

Park's discovery is only partially reported in [Par81a]. The main topic of that paper is a different one, namely omega-regular languages (extensions of regular languages containing also infinite sequences) and operators for fair concurrency. And the main contributions, as claimed by Park, are: characterisations of these languages as sets of recursive equations involving minimal and maximal fixed points, something fairly novel at the time (Park in fact had already proposed similar constructions in previous years, see for instance [Par79]); characterisations of these languages as the languages accepted by certain variants of Büchi automata (the main difference over Büchi automata is that Park's automata also recognise finite words, but this does not affect much of the theory of the automata); properties of closure of the languages under a fair-concurrency operator.

Bisimulation appears at the end, as a secondary contribution, as a proof technique for trace equivalence on automata. Park himself does not seem to believe much in it. He writes, for instance:

> This provides motivation to be interested in proof principles for automata such as those involved here [cf: bisimulation] even though their utility for the purposes of operational semantics of programs is obviously limited.

Bisimulation is first given on finite automata, but only as a way of introducing the concept on the Büchi-like automata investigated in the paper. Here, bisimulation has additional clauses that make it non-transitive and different from the definition of bisimulation we know today. Further, bisimilarity and the coinduction proof method are not mentioned in the paper.

Indeed, Park never writes a paper to report on his findings about bisimulation. It is possible that this does not appear to him a contribution important enough to warrant a paper: he considers bisimulation a variant of the earlier notion of simulation by Milner [Mil70, Mil71b]; and it is not in Park's style to write many papers. The best account I have found of Park's discovery of bisimulation are the summary and the slides of his talk at the 1981 Workshop on the Semantics of Programming Languages [Par81b].

4.4 Discussion

In Computer Science, the move from homomorphism to bisimulation follows somehow an opposite path with respect to Modal Logic: first homomorphisms are made relational, then they are made symmetric, by adding a reverse implication.

It remains puzzling to me why bisimulation has been discovered so late in Computer Science. For instance, in the 1960s weak homomorphism is well-known in automata theory and, as discussed in Section 4.1, this notion is not that far from simulation. Another emblematic example, again from automata theory, is given by the algorithm for minimisation of deterministic automata, already known in the 1950s [Huf54, Moo56] (also related to this is the Myhill-Nerode theorem [Ner58]). The aim of the algorithm is to find an automaton equivalent to a given one but minimal in the number of states. The algorithm proceeds by progressively constructing a relation $\mathcal{S}$ with all pairs of non-equivalent states. It roughly goes as follows. First step (a) below is applied, to initialise $\mathcal{S}$; then step (b), where new pairs are added to $\mathcal{S}$, is iterated until a fixed point is reached, i.e., no further pairs can be added.

a. For all states $s, t,$: if s final and t is not, or vice versa, then $s \,\mathcal{S}\, t$

b. For all states s, t, such that $\neg(s \,\mathcal{S}\, t)$: if there is a such that $\mathcal{T}_a(s) \,\mathcal{S}\, \mathcal{T}_a(t)$ then $s \,\mathcal{S}\, t$

The final relation gives all pairs of non-equivalent states. Then its complement, say $\overline{\mathcal{S}}$, gives the equivalent states. In the minimal automaton, the states in the same equivalence class for $\overline{\mathcal{S}}$ are collapsed into a single state.

The algorithm strongly reminds us of the Paige-Tarjan's partition refinement algorithm [PT87], the best known algorithm for computing bisimilarity and for minimisation modulo bisimilarity. Indeed, the complement relation $\overline{\mathcal{S}}$ that one wants to find has a natural coinductive definition, as a form of bisimilarity, namely the largest relation $\mathcal{R}$ such that

1. if $s \,\mathcal{R}\, t$ then either both s and t are final or neither is;
2. for each a, if $s \,\mathcal{R}\, t$ then $\mathcal{T}_a(s) \,\mathcal{R}\, \mathcal{T}_a(t)$.

Further, any relation $\mathcal{R}$ that satisfies the conditions (1) and (2)—that is, any bisimulation—only relates pairs of equivalent states and can therefore be used to determine equivalence of specific states.

The above definitions and algorithm are for deterministic automata. Bisimulation would have been interesting also on non-deterministic automata. Although on such automata bisimilarity does not coincide with trace equivalence—the standard equality on automata—at least bisimilarity implies trace equivalence and the algorithms for bisimilarity have a better complexity (P-complete [ABGS91, BGS92], rather than PSPACE-complete [MS72, KS90]).

5 Set Theory

In Mathematics, bisimulation and concepts similar to bisimulation are formulated in the study of properties of extensionality of models. Extensionality guarantees that equal objects cannot be distinguished within the given model. When the structure of the objects, or the way in which the objects are supposed to be used, are non-trivial, the "correct" notion of equality may be non-obvious. This is certainly the case for non-well-founded sets, as they are objects with an infinite depth, and indeed most of the developments in Set Theory towards bisimulation are made in a line of work on the foundations of theories of non-well-founded sets. Bisimulation is derived from the notion of isomorphism (and homomorphism), intuitively with the objective of obtaining relations coarser than isomorphism but still with the guarantee that related sets have "the same" internal structure.

Bisimulation is first introduced by Forti and Honsell and, independently, by Hinnion, around the same time (beginning of the 1980s). It is recognised and becomes important with the work of Aczel and Barwise. Some earlier constructions, however, have a clear bisimulation flavor, notably Mirimanoff's isomorphism at the beginning of the 20th century.

5.1 Non-well-founded sets

Non-well-founded sets are, intuitively, sets that are allowed to contain themselves. As such they violate the *axiom of foundation*, according to which the membership relation on sets does not give rise to infinite descending sequences

$$\ldots A_n \in A_{n-1} \in \ldots \in A_1 \in A_0 \ .$$

For instance, a set Ω which satisfies the equation $\Omega = \{\Omega\}$ is circular and as such non-well-founded. A set can also be non-well-founded without being circular; this can happen if there is an infinite membership chain through a sequence of sets all different from each other.

If the axiom of foundation is used, the sets are *well-founded*. On well-founded sets the notion of equality is expressed by Zermelo's *extensionality axiom*: two sets are equal if they have exactly the same elements. In other words, a set is precisely determined by its elements. This is very intuitive and naturally allows us to reason on equality proceeding by (transfinite) induction on the membership relation. For instance, we can thus establish that the relation of equality is unique. Non-well-founded sets, by contrast, may be infinite in depth, and therefore inductive arguments may not be applicable. For instance, consider the sets A and B defined via the equations $A = \{B\}$ and $B = \{A\}$. If we try to establish that they are equal via the extensionality axiom we end up with a tautology ("A and B are equal iff A and B are equal") that takes us nowhere.

Different formulations of equality on non-well-founded sets appear during the 20th century, together with proposals for *axioms of anti-foundation*.

5.2 The stratified approach to set theory

The first axiomatisation of set theory by Ernst Zermelo in 1908 [Zer08] has seven axioms, among which is the axiom of extensionality. However, it has no axioms of foundation, and the possibility of having circular sets is in fact left open (page 263, *op. cit.*).

In the same years, Bertrand Russell strongly rejects all definitions that can involve forms of circularity ("whatever involves all of a collection must not be one of the collection", in one of Russell's formulations [Rus08]). He favors a *theory of types* that only allows *stratified* constructions, where objects are hereditarily constructed, starting from atoms or primitive objects at the bottom and then iteratively moving upward through the composite objects. A preliminary version of the theory is announced by Russell already in 1903 [Rus03, Appendix B]; more complete and mature treatments appear in 1908 [Rus08] and later, in 1910, 1912, 1913, in the monumental work with Alfred North Whitehead [RW13].

Russell's approach is followed by the main logicians of the first half of the 20th century, including Zermelo himself, Abraham Fraenkel, Thoralf Skolem, Johann von Neumann, Kurt Gödel, Paul Bernays. Their major achievements include the formulation of the axiom of foundation, and the proofs of its consistency and independence. An axiom of foundation is deemed necessary so to have a "canonical" universe of sets. Without foundation, different interpretations are possible, some including circular sets. This possibility is clearly pointed out as a weakness by Skolem [Sko23], and by Fraenkel [Fra22], where circular sets (precisely, Mirimanoff's "ensembles extraordinaires", see below) are labelled as "superfluous". It will be formally proved by Bernays only in 1954 [Ber54] that the existence of circular sets does not lead to contradictions in the Zermelo-Fraenkel system without axiom of foundation.

Remark 5.1 The axiom of foundation forces the universe of sets in which the other axioms (the basic axioms) should be interpreted to be the smallest possible one; i.e., to be an "inductive universe". By contrast, axioms of anti-foundation lead to the largest possible universe, i.e., a "coinductive universe". Indeed, referring to the algebraic/coalgebraic interpretation of induction/coinduction, the foundation axiom can be expressed as a requirement that the universe of sets should be an *initial algebra* for the powerset functor, whereas anti-foundation (as in Forti and Honsell, Aczel, and Barwise) can be expressed as a requirement that the universe should be a *final coalgebra* for the same functor. The former is an inductive definition of the universe, whereas the latter is a coinductive one. □

The motivations for formalising and studying the stratified approach advocated by Russell were strong at the beginning of the 20th century. The discovery of paradoxes such as Burali-Forti's and Russell's had made the set theory studied by Cantor and Frege shaky, and circularity—with no distinction of cases—was generally perceived as the culprit for these as well as for paradoxes known in other fields. Further, the stratified approach was in line with common sense and

perception (very important in Russell's conception of science), which denies the existence of circular objects.

The stratified approach remains indeed *the only* approach considered (in Logics and Set Theory), up to roughly the 1960s; with the exception of Mirimanoff and Finsler that we discuss below. The stratified approach has also inspired—both in the name and in the method—type theory in Computer Science, notably in the works of Church, Scott, and Martin-Löf. It will be first disputed by Jean-Yves Girard and John Reynolds, in the 1970s, with the introduction of impredicative polymorphism.

5.3 Non-well-founded sets and extensionality

Dimitry Mirimanoff first introduces in 1917 the distinction between well-founded and non-well-founded sets, the "ensembles *ordinaires* et *extraordinaires*" in Mirimanoff's words [Mir17a] (on the same topic are also the two successive papers [Mir17b] and [Mir20]). Mirimanoff realises that Zermelo's set theory admitted sophisticated patterns of non-well-foundedness, beyond the "simple" circularities given by self-membership as in the purely reflexive set $\Omega = \{\Omega\}$. In [Mir17b], Mirimanoff also tries to give an intuition for the non-well-founded sets; he recalls the cover of a children book he had seen, with the image of two children looking at the cover of a book, which in turn had the image of two children, in a supposedly infinite chain of nested images.

Mirimanoff defines an interesting notion of isomorphism between sets, that we report in Section 5.8. Mirimanoff does not however go as far as proposing an axiom of extensionality more powerful than Zermelo's. This is first attempted by Paul Finsler, in 1926 [Fin26]. Finsler presents 3 axioms for a universe of sets equipped with the membership relation. The second one is an extensionality axiom, stipulating that isomorphic sets are equal. Finsler's notion of isomorphism between two sets X and Y—which is different from Mirimanoff's—is, approximately, a bijection between the transitive closures of X and Y (more precisely, the transitive closures of the unit sets $\{X\}$ and $\{Y\}$; the precise meaning of isomorphism for Finsler can actually be debated, for it appears in different forms in his works).[4] Finsler uses graph theory to explain properties and structure of sets, something that later Aczel will make more rigorous and at the heart of his theory of non-well-founded sets.

Mirimanoff and Finsler's works are remarkable: they go against the standard approach to set theory at the time; and against the common sense according to which objects are stratified and circular sets are "paradoxical". For Mirimanoff and Finsler, not all circular definitions are dangerous, and it is a task for the logicians to isolate the "good" ones.

The attempts by Mirimanoff and Finsler remain little known. We have to

[4] A set A is transitive if each set B that is an element of A has the property that all the elements of B also belong to A; that is, all composite elements of A are also subsets of A. The transitive closure of a set C is the smallest transitive set that contains C. Given C, its transitive closure is intuitively obtained by copying at the top level all sets that are elements of C, and then recursively continuing so with the new top-level sets.

wait till around the 1960s with, e.g., Specker [Spe57] and Scott [Sco60], to see a timid revival of the interest in non-well-founded structures, and the late 1960s, and then the 1970s and 1980s, for a wider revival, with Boffa (with a number of papers, including [Bof68, Bof69, Bof72]) and many others. New proposals for anti-foundation axioms are thus made, and with them, new interpretations of extensionality on non-well-founded sets, notably from Scott [Sco60], and Forti and Honsell [FH83]. Forti and Honsell obtain bisimulation; their work is then developed by Aczel and Barwise. We discuss Forti and Honsell, Aczel, and Barwise's contributions below. On the history of non-well-founded sets, the reader may also consult Aczel [Acz88, Appendix A].

5.4 Marco Forti and Furio Honsell

Marco Forti and Furio Honsell's work on non-well-founded sets [Hon81, FH83] (and various papers thereafter) is spurred by Ennio De Giorgi, a well-known analyst who, in the 1970s and 1980s, organises regular weekly meetings at the Scuola Normale Superiore di Pisa, on logics and foundations of Mathematics. In some of these meetings, De Giorgi proposes constructions that could yield infinite descending chains of membership on sets, that Forti and Honsell then go on to elaborate and develop.

The most important paper is [FH83]. Here Forti and Honsell study a number of anti-foundation axioms, derived from a "Free Construction Principle" proposed by De Giorgi. They include axioms that already appeared in the literature (such as Scott's [Sco60]), and a new one, called X_1, that gives the strongest extensionality properties, in the sense that it equates more sets. (We recall X_1 in the next section, together with Aczel's version of it.) The main objective of the paper is to compare the axioms, and define models that prove their consistency. Bisimulations and similar relations are used in the constructions to guarantee the extensionality of the models.

Forti and Honsell use, in their formulation of bisimulation, functions $f : A \mapsto \wp(A)$ from a set A to its powerset $\wp(A)$. Bisimulations are called *f-conservative* relations and are defined along the lines of the fixed-point interpretation of bisimulation in Section 2.3. We can make a state-transition interpretation of their definitions, for a comparison with today's definition (Definition 2.3). If f is the function from A to $\wp(A)$ in question, then we can think of A as the set of the possible states, and of f itself as the (unlabeled) transition function; so that $f(x)$ indicates the set of possible "next states" for x. Forti and Honsell define the fixed point behaviour of f on the relations on A, via the functional F defined as follows[5]. If $\mathcal{R}$ is a relation on A, and $s, t \in A$, then $(s, t) \in F(\mathcal{R})$ if:

- for all $s' \in f(s)$ there is $t' \in f(t)$ such that $s' \mathcal{R} t'$;
- the converse, i.e. for all $t' \in f(t)$ there is $s' \in f(s)$ such that $s' \mathcal{R} t'$.

[5] We use a notation different from Forti and Honsell here.

Figure 3: Sets as graphs

A reflexive and symmetric relation $\mathcal{R}$ is f-conservative if $\mathcal{R} \subseteq F(\mathcal{R})$; it is f-admissible if it is a fixed point of F, i.e., $\mathcal{R} = F(\mathcal{R})$. The authors note that F is monotone over a complete lattice, hence it has a greatest fixed point (the largest f-admissible relation). They also prove that such greatest fixed point can be obtained as the union over all f-conservative relations (the coinduction proof principle), and also, inductively, as the limit of a sequence of decreasing relations over the ordinals that starts with the universal relation $A \times A$ (akin to the characterisation in Theorem 2.12). The main difference between f-conservative relations and today's bisimulations is that the former are required to be reflexive and symmetric.

However, while the bisimulation proof method is introduced, as derived from the theory of fixed points, it remains rather hidden in Forti and Honsell's works, whose main goal is to prove the consistency of anti-foundation axioms. For this the main technique uses the f-admissible relations.

5.5 Peter Aczel

In Mathematics, bisimulation and non-well-founded sets are made popular by Peter Aczel, notably with his book [Acz88]. Aczel is looking for mathematical foundations of processes, prompted by the work of Milner on CCS and his way of equating processes with an infinite behaviour via a bisimulation quotient. Aczel reformulates Forti and Honsell's anti-foundation axiom X_1. In Forti and Honsell [FH83], the axiom says that from every relational structure there is a unique homomorphism onto a transitive set. Aczel calls the axiom AFA and expresses it with the help of graph theory, in terms of graphs whose nodes are decorated with sets. For this, sets are thought of as (pointed) graphs, where the nodes represent sets, the edges represent the converse membership relation (e.g., an edge from a node x to a node y indicates that the set represented by y is a member of the set represented by x), and the root of the graph indicates the starting point, that is, the node that represents the set under consideration. For instance, the sets $\{\emptyset, \{\emptyset\}\}$ and $D = \{\emptyset, \{D\}\}$ naturally corresponds to the graphs of Figure 3 (where for convenience nodes are named) with nodes 2 and c being the roots. The graphs for the well-founded sets are those without infinite paths or cycles, such as the graph on the left in Figure 3. AFA essentially states that each graph represents a unique set. This is formalised via the notion of *decoration*. A decoration for a graph is an assignment of sets to nodes that

respects the structure of the edges; that is, the set assigned to a node is equal to the set of the sets assigned to the children of the node. For instance, the decoration for the graph on the left of Figure 3 assigns $\emptyset$ to node 0, $\{\emptyset\}$ to node 1, and $\{\emptyset, \{\emptyset\}\}$ to node 2, whereas that for the graph on the right assigns $\emptyset$ to a, $\{D\}$ to b, and $\{\emptyset, \{D\}\}$ to c. Axiom AFA stipulates that *every graph has a unique decoration.* (In Aczel, the graph plays the role of the relational structure in Forti and Honsell, and the decoration the role of the homomorphism into a transitive set.) In this, there are two important facts: the existence of the decoration, and its uniqueness. The former tells us that the non-well-founded sets we need do exist. The latter tell us what is equality for them. Thus two sets are equal if they can be assigned to the same node of a graph. For instance the sets Ω, A and B in Section 5.1 are equal because the graph

has a decoration in which both nodes receive Ω, and another decoration in which the node on the left receives A and that on the right B. Bisimulation comes out when one tries to extract the meaning of equality. A bisimulation relates sets A and B such that

- for all $A_1 \in A$ there is $B_1 \in B$ with A_1 and B_1 related; and the converse, for the elements of B_1.

Two sets are equal precisely if there is a bisimulation relating them. The bisimulation proof method can then be used to prove equalities between sets, for instance the equality between the sets A and B above. This equality among sets is also referred to as *strong extensionality* because it is the most generous, or the coarsest, equality that is compatible with the membership structure of sets: two sets are different only if there they present some genuine, observable, structural difference.

Aczel formulates AFA towards end 1983; he does not publish it immediately having then discovered the earlier work of Forti and Honsell and the equivalence between AFA and X_1. Instead, he goes on developing the theory of non-well-founded sets, mostly through a series of lectures in Stanford between January and March '85, which leads to the book [Acz88]. Aczel shows how to use the bisimulation proof method to prove equalities between non-well-founded sets, and develops a theory of coinduction that sets the basis for the coalgebraic approach to semantics (Final Semantics).

Up to Aczel's book [Acz88], all the works on non-well-founded sets had remained outside the mainstream. This changes with Aczel, for two main reasons: the elegant theory that he develops, and the concrete motivations for studying non-well-founded sets that he brings up.

Something that influences the developments of non-well-founded sets, and that is manifest in Aczel's work is *Mostowski's Collapse* Lemma (proved probably sometime in the 1940s and today recognised as fundamental in the study of models of set theory). The original statement of the lemma talks about well-founded relations; roughly it says that given any such relation there is a

unique set that faithfully represents the relation in its membership structure. Aczel reformulates the collapse on graphs. It becomes the assertion that every well-founded graph has a unique decoration. Axiom AFA is then obtained by removing the well-foundedness hypothesis (of course now, on the non-well-founded sets, it is an axiom, whereas Mostowski's collapse on the well-founded sets is a lemma). The collapse is also fundamental for the formal representation of sets as graphs, as it allows us to conclude that we can associate a unique set to each pointed graph, via the decoration. When Finsler writes his 1926 paper [Fin26], the collapse construction is not known and indeed Finsler's use of graph theory remains informal.

5.6 Jon Barwise

Aczel's original motivation for the study on non-well-founded sets is to provide set-theoretic models for CCS. Jon Barwise brings up other applications, notably the study of paradoxes such as the Liar paradox in Philosophical Logic and more broadly the study of meaning in natural (i.e., human spoken) languages [BE87].

Further, Barwise develops a theory of non-well-founded sets that is not based on the relationship between sets and graph theory as Aczel, but, instead, on systems of equations. The axiom AFA becomes a requirement that appropriate systems of equations have a unique solution. To understand this point consider that, as the purely reflexive set Ω can be seen as the solution to the equation $x = \{x\}$, so all non-well-founded sets arise from systems of equations with variables on the left-hand side, and well-founded sets possibly containing such variables on the right-hand side. In Aczel [Acz88] this is expressed as the Solution Lemma. Barwise makes it the base assumption from which all the theory of sets is derived. For more details, the reader may consult Barwise's book with Lawrence Moss [BM96].

5.7 Extensionality quotients: Roland Hinnion and others

More or less at the same time as Forti and Honsell, and independently from them, bisimulation-like relations are used by Roland Hinnion [Hin80, Hin81] (a related, but later, paper is also [Hin86]). Hinnion follows Mostowski's collapse; Mostowski's construction allows one to obtain, from a well-founded relational structure, a model of set theory in which Zermelo's axiom of extensionality holds. Hinnion aims at generalising this to arbitrary (i.e., not necessarily well-founded) structures. Thus he defines forms of bisimulation on relational structures, as usual the "transitions" of the bisimulation game being dictated by the relation on the structure. He considers two such forms. The *final equivalences* (later [Hin86] called *increasing*) are the bisimulations that are also equivalences. The *contractions* are the final equivalences whose quotient on the original structure satisfies the extensionality axiom. Roughly we can think of contractions as bisimulation equivalences that are also congruences, in that they are preserved by the operators of the model, i.e, by the addition of external structure.

Hinnion does not formulate axioms of anti-foundation. Thus while imposing the AFA axiom makes equality the only possible bisimulation for any structure, Hinnion uses bisimulations to define new structures, via a quotient.

Although Hinnion points out that final equivalences and contractions form a complete lattice, he does not put emphasis on the maximal ones. The equalities obtained via his quotients can indeed be finer than the equality that the AFA axiom yields (which corresponds to the quotient with bisimilarity, the maximal bisimulation). Consequently, he also does not put emphasis on the coinduction proof principle associated to bisimulation.

Constructions similar to Hinnion's, that is, uses of relations akin to bisimulation to obtain extensional quotient models, also appear in works by Harvey Friedman [Fri73] and Lev Gordeev [Gor82]. In this respect, however, the first appearance of a bisimulation relation I have seen is in a work by Jon Barwise, Robin O. Gandy, and Yiannis N. Moschovakis [BGM71], and used in the main result about the characterisation of the the structure of the next admissible set A^+ over a given set A. (Admissible Sets form a Set Theory weaker than Zermelo-Fraenkel's in the principles of set existence; it was introduced in the mid 1960s by Saul Kripke and Richard Platek with the goal of generalising ordinary recursion theory on the integers to ordinals smaller than a given "well-behaved" one.) A stronger version of the result is found in Moschovakis's book [Mos74] (where the main result is Theorem 9E.1, in Chapter 9, and the bisimulation relation is used in the proof of Lemma 9). As most of the results we have mentioned in Set Theory, so the Barwise-Gandy-Moschovakis Theorem is inspired by Mostowski's Collapse Lemma. While the papers [BGM71, Fri73, Gor82] make use of specific bisimulation-like relations, they do not isolate or study the concept.

5.8 Discussion

It may appear surprising that also in Mathematics it takes so long for the notion of bisimulation to appear. This is partly explained by the limited attention to non-well-founded structures up-to the 1980s, as discussed in Section 5.2.

It is fair to say, however, that some of the very early constructions had already a definite bisimulation flavor. An enlightening example is Mirimanoff's pioneering work on non-well-founded sets. Mirimanoff [Mir17a] defines a notion of isomorphism for sets that have atoms (often called urelements), i.e., elements that cannot be decomposed and that are not the empty set. Two such sets E and E' are deemed *isomorphic* when the two conditions below hold:

1. The sets E and E' are equivalent; that is, a perfect correspondence can be established between the elements of E and E'

2. Further, the above correspondence can be established in such a way that each atom e in E corresponds to an atom e' in E' and conversely; and each element-set F of E corresponds to an element-set F' of E' (an element-set of a set G is an element of G that is a set). The perfect correspondence between F and F' can then be established in a way that each atom in F

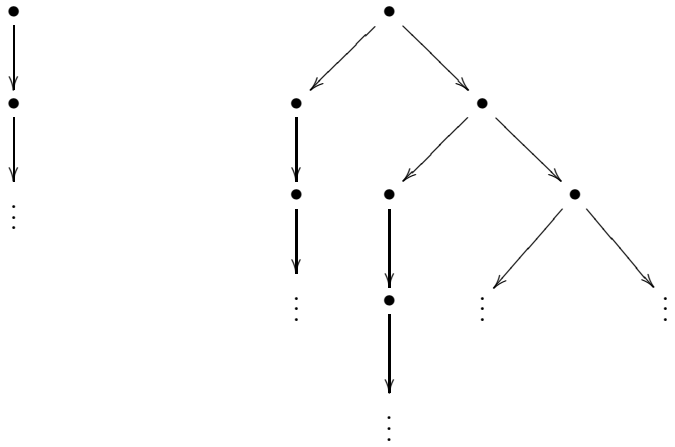

Figure 4: Tree unfolding of the sets Ω and U

---

corresponds to an atom in F', and each element-set of F corresponds to an element-set of F'; and so forth.

Although today we would give this definition in a different way, its meaning is clear. Mirimanoff's isomorphism abstracts from the nature and identity of the atoms. His intention—clearly stated in [Mir17b]—is to relate sets with the same tree structure, as determined by the membership relation. (In other words, if we think of sets as trees, along the lines of the representation of sets as graphs mentioned in Section 5.5, then Mirimanoff's isomorphism is essentially an isomorphism between such trees.)

Mirimanoff's isomorphism is not far from the equality on sets given by Finsler's and Scott's anti-foundation axioms. These equalities too, indeed, are based on notions of isomorphism. The peculiarity of Mirimanoff's definition is that it is built on the idea of equating potentially infinite objects by decomposing, or observing, them top-down, from a composite object to its constituents. This idea is also at the heart of the definition of bisimulation (where, for instance, decomposing a process is observing its transitions). The "perfect correspondence" used by Mirimanoff is however a bijection between the components of the sets, rather than a relation, and as such the resulting notion of equality is finer than bisimilarity. For instance, consider the purely reflexive set Ω and the set $U = \{\Omega, U\}$. It is easy to see that they are bisimilar. However they are not isomorphic for Mirimanoff as their trees, in Figure 4, are quite different. (The two sets are also different under Finsler's and Scott's equalities; bisimilarity is indeed strictly coarser than Finsler's and Scott's equalities, see [Acz88].)

What really makes Mirimanoff's isomorphism different from bisimulation is that Mirimanoff fails to promote isomorphism to equality for sets. For instance

the sets $A = \{B\}$ and $B = \{A\}$ are isomorphic but not equal, hence the set $\{A, B\}$ has two elements and is not isomorphic to the set $\{A\}$ or to the set $\{B\}$, which have only one element. To identify isomorphism and equality, the clause of isomorphism, in establishing the "perfect correspondence" between the elements of two isomorphic sets, should take into account the collapse given by isomorphism itself. This can be easily obtained by weakening the requirements of injectivity and surjectivity in the correspondences, for instance making the correspondences total relations—i.e., making them bisimulations. In conclusion, had Mirimanoff investigated the impact of isomorphism on extensionality, while retaining the spirit of his definition, he would have probably discovered bisimulation.

However, Mirimanoff pays little attention to extensionality. His main interest, motivated by Burali-Forti and Russell's paradoxes, is understanding what are the conditions for the existence of a set of objects. And his main results are theorems asserting—using modern terminology—that certain classes are not sets. In Set Theory, even more than in Modal Logic or Computer Science, the move from the "functional" concepts of homomorphism and isomorphism to the "relational" concept of bisimulation will take time.

6 The introduction of fixed points in Computer Science

Bisimulation and the bisimulation proof method, as coinductive concepts, are intimately related to fixed points. We therefore also examine coinduction and fixed points. We do not attempt, however, to trace the general history of fixed-point theory—in Mathematics this is a story stretching far back in time. Instead, we concentrate on Computer Science, and recall some papers that well witness the introduction of coinduction and fixed-point theory for the design and analysis of programming languages. Knaster-Tarski Theorem 2.9, about the existence of least and greatest fixed point for a monotone function on a complete lattice, or variations of this such as Theorem 2.11, are the starting point for all the works we mention.

The earliest uses of fixed points in Computer Science, in the form of least fixed points, can be found in: recursive function theory, see for instance Rogers's book [Rog67] and references therein; formal language theory, as in the work of Arden [Ard60] and Ginsburg and Rice [GR62]. However, distinguishing Computer Science from recursive function theory, the importance of fixed points in Computer Science really comes up only at the end of the 1960s, with four independent papers, roughly at the same time, by Dana Scott and Jaco de Bakker [SdB69], Hans Bekič [Bek69], David Park [Par69], and Antoni Muzurkiewicz [Maz71] (however [Maz71] does not make explicit reference to fixed-point theory). Although [Maz71] is published in 1971, it is already made available, as a working paper, in December 1969 to the IFIP Working Group 2.2, whose members included some of the most influential researchers on programming language

concepts at that time; this paper also had an impact on the discovery of continuations in denotational semantics, see [Rey93]. It might sound surprising that [SdB69] and [Bek69] should be considered "independent", given that both appear as manuscripts from the same place, the Vienna IBM Laboratory. The reason is that Bekič's work is mainly carried out during a one-year visit (November 1968–November 1969) at Queen Mary College, London, where Bekič stays in Peter Landin's group (indeed Landin has a strong influence on [Bek69]). Thus when Scott and de Bakker's work is presented at the Vienna IBM Laboratory in August 1969, Bekič—who is a member of the Laboratory—is still in London. The first time when the two works can be discussed and compared is the IFIP Working Group 2.2 meeting in Colchester in September 1969. (Indeed, if we were to fix a date and a place for the introduction of fixed points in Computer Science it should probably be those of this meeting.)

The above four papers bring out the importance of least fixed points for the semantics of programs, the relevance of lattice theory and the Knaster-Tarski Theorem 2.9, and propose various rules for reasoning about least fixed points. Programs take the form of recursive function definitions or of flowcharts. Further, [SdB69] paves the way for the fundamental work on denotational semantics by Scott and Strachey in Oxford in the 1970s, where least fixed points, and continuity of functions, are essential. Influential on the above four papers are earlier works on program correctness and on uses of the "paradoxical combinator" Y of the λ-calculus, notably papers by Landin such as [Lan64], by McCarthy such as [McC61, McC63], and by Floyd such as [Flo67]. For instance, McCarthy [McC61, McC63] proposes the first method for proving properties of recursive programs, called *recursion induction*; variants and stronger versions of the method are formulated in [SdB69], [Bek69], and [Par69]. Also, the fixed-point properties of the Y combinator of the λ-calculus had been known for a long time (used for instance by Curry, Feys, Landin, and Strachey), but the precise mathematical meaning of Y as fixed point remains unclear until Scott works out his theory of reflexive domains, at the end of 1969 [Sco69b, Sco69a]; see [Par70]. (Another relevant paper is [Sco69c], in which fixed-points appear but which preceds the discovery of reflexive domains. We may also recall James H. Morris, who earlier [Mor68] had proved a minimal fixed-point property for the Y combinator; in the same document, Morris had considered the relationship between least fixed points and functions computed by first-order recursive definitions of programs.)

During the 1970s, further fixed-point techniques and rules are put forward. A number of results on fixed points and induction rules, and the basic theory of continuous functions, are due to Scott, e.g. [Sco72b, Sco72a, Sco76]. On uses of least fixed points in semantics and in techniques for program correctness, we should also mention the work of de Bakker and his colleagues in The Netherlands, e.g. [BR73, Bak75, Bak71]; the Polish school, with Mazurkiewicz, Blikle, and colleagues, e.g., [Maz71, Maz73, Bli77]; the work of Ugo Montanari and colleagues in Pisa, such as [GM76] that contains notions of observations and equivalence of representations in abstract data types that today we recognize as related to fixed points via the concept of finality. Other references to the early

works on fixed points can be found in Zohar Manna's textbook [Man74].

The above works all deal with least fixed points. Greatest fixed points, and related coinductive techniques, begin to appear as well in the 1970s. It is hard to tell what is the first appearance. One reason for this is that the rules for greatest fixed points are not surprising, being the dual of rules for least fixed points that had already been studied. I would think however that the first to make explicit and non-trivial use of greatest fixed points is David Park, who, throughout the 1970s, works intensively on fairness issues for programs that may contain constructs for parallelism and that may not terminate. The fixed-point techniques he uses are rather sophisticated, possibly involving alternation of least and greatest fixed points. Park discusses his findings in several public presentations. A late overview paper is [Par79]; we already pointed out in Section 4.3 that Park did not publish much.

Other early interesting uses of greatest fixed points are made by the following authors. Mazurkiewicz [Maz73] studies properties of computations from processes, where processes are modelled via forms of LTSs; the properties studied include divergence and termination. The way in which Mazurkiewicz defines divergent states (i.e., the states from which a computation may not terminate) and the technique proposed to prove divergence of states are coinductive, though—as in his earlier paper [Maz71]—there is no explicit reference to fixed-point theory.

Edmund Clarke [Cla77] shows that the correctness proofs for Floyd-Hoare axiom systems—deductive systems for partial correctness based on invariance assertions intensively investigated in the 1970s—could be elegantly formalised by means of fixed-point theory, whereby: program invariants become greatest fixed points; completeness of a system becomes the proof of the existence of a fixed point for an appropriate functional; and soundness is derived from the maximality of such fixed point. Thus soundness is a coinductive proof. Willem-Paul de Roever [dR77] strongly advocates the coinduction principle as a proof technique (he calls it "greatest fixed point induction"). De Roever uses the technique to reason about divergence, bringing up the duality between this technique and inductive techniques that had been proposed previously to reason on programs.

Coinduction and greatest fixed points are implicit in a number of earlier works in the 1960s and 1970s. Important examples, with a huge literature, are the works on unification, for instance on structural equivalence of graphs, and the works on invariance properties of programs. Fixed points are also central in *stream processing* systems (including *data flow* systems). The introduction of streams in Computer Science is usually attributed to Peter Landin, in the early 1960s (see [Lan65a, Lan65b] where Landin discusses the semantics of Algol 60 as a mapping into a language based on the λ-calculus and Landin's SECD machine [Lan64], and historical remarks in [Bur75]). However, fixed points are explicitly used to describe stream computations only after Scott's theory of domain, with the work of Gilles Kahn [Kah74].

I do not know when and who first used the word "coinduction". The first appearance of the term I am aware of is in Barwise and Etchemendy's 1987 book [BE87]; I am therefore led to think that the term has been introduced by

Barwise. The term "coinduction" also occurs in earlier papers and textbooks in Mathematics (e.g., [Mos74]), but simply to indicate the complement of an inductively-defined structure. In Computer Science, the term is widely used after Milner and Tofte [MT91], who use coinduction to prove the soundness of a type system, and describe coinduction to explain the analogy between the technique for types in their paper and the bisimulation techniques. The main objective of the paper is indeed to advocate the proof technique and to suggest the name coinduction for it. At the time of the writing, the authors of [MT91] were not aware of the other occurrences of the name.[6] Given the duality with induction, the term "coinduction" is so natural to appear a kind of folklore term, and this makes it hard, today, to trace back its introduction.

7 Fixed-point theorems

We conclude with a few remarks on the fixed-point theorems of Section 2.3. Theorem 2.9 is usually called the "Knaster-Tarski fixed-point theorem". The result was actually obtained by Tarski, who, in a footnote to [Tar55] (footnote 2, page 286), where the result appears as Theorem 1, explains its genesis:

> In 1927 Knaster and the author [i.e., Tarski] proved a set-theoretical fixed point theorem by which every function on and to the family of all subsets of a set, which is increasing under set-theoretical inclusion has at least one fixed point; see [Kna28] where some applications of this result in set theory [...] and topology are also mentioned. A generalisation of this result is the lattice-theoretical fixed point theorem stated above as Theorem 1. The theorem in its present form and its various applications and extensions were found by the author in 1939 and discussed it in a few public lectures in 1939–1942. (See, for example, a reference in the American Mathematical Monthly 49(1942), 402.) An essential part of Theorem 1 was included in [Bir48, p. 54]; however the author was informed by Professor Garret Birkhoff that a proper historical reference to this result was omitted by mistake.

Tarski first properly publishes the theorem in 1955 [Tar55], together with a few related results, proofs, and applications to boolean algebras and topology. He had anticipated a summary of [Tar55] in 1949, as [Tar49]. Credit for the theorem is also given to Bronislaw Knaster because of the following result in [Kna28]:

Lemma 7.1 *If F is monotone function over sets such that there is a set W with $F(W) \subseteq W$ then there is a subset $\mathcal{Q}$ of W such that $F(\mathcal{Q}) = \mathcal{Q}$.*

[6] Personal communications with R. Milner and M. Tofte; the idea of using the name "coinduction" came up to them most likely sometime between 11 February 1988 and 17 March 1988.

While Theorem 2.9 describes the structure of the fixed points of a function, Lemma 7.1 only asserts the existence of at least one fixed point. Moreover Theorem 2.9 is on arbitrary lattices, whereas we can think of Lemma 7.1 as being on the special lattices given by the powerset construction. [Kna28] is actually a very short note—about one page—in which the lemma is used to derive, as a special case, a theorem on monotone functions over sets. The note itself confirms that the results presented had been obtained by Knaster together with Tarski.

It would not be so surprising if Theorem 2.9 had been obtained around the same period also by Stephen C. Kleene or Leonid V. Kantorovich[7], although the writings from these authors that we have today only deal with constructive proofs of the existence of least and greatest fixed points along the lines of Theorem 2.11 (see below).

In Computer Science, Theorem 2.11 is often called the "Kleene fixed-point theorem", with reference to Kleene's first recursion theorem [Kle52], and often presented on pointed complete partial orders and for least fixed points. The first recursion theorem is obtained by Kleene around the end of the 1930s, as reported in [Kle52, Kle70]. Around that time, or anyhow before the 1950s, Theorem 2.11 is independently known to other authors, first of all Tarski and Kantorovich (for instance Theorem I in [Kan39] is similar to Theorem 2.11), but possibly others—see also the discussion in [CC79, page 56]. It is indeed unclear who should be credited for the theorem. Lassez, Nguyen, and Sonenberg [LNS82] consider the origins of this theorem (as well as of the other fixed-point theorems) and conclude that it should be regarded as a "folk theorem".

Theorem 2.12 is from Hitchcock and Park [HP73]. Similar versions are also given by Devidé [Dev63], Pasini [Pas74], Cadiou [Cad72], Cousot and Cousot [CC79]. A related theorem also appears in Bourbaki [Bou50].

References

[ABGS91] Carme Alvarez, José L. Balcázar, Joaquim Gabarró, and Miklos Santha. Parallel complexity in the design and analysis on conurrent systems. In *Proc. PARLE '91: Parallel Architectures and Languages Europe, Volume I: Parallel Architectures and Algorithms*, volume 505 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 1991.

[AC93] R. M. Amadio and L. Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems*, 15(4):575–631, 1993. A preliminary version appeared in POPL '91 (pp. 104–118), and as DEC Systems Research Center Research Report number 62, August 1990.

[7] Leonid Vitalyevich Kantorovich (1912–1986) was a Russian mathematician who obtained the Nobel Prize for Economics in 1975 for his work on the allocation and optimisation of resources, in which he pioneers the technique of linear programming.

[Acz88] P. Aczel. *Non-well-founded Sets*. CSLI lecture notes, no. 14, 1988.

[Acz93] P. Aczel. Final universes of processes. In Brookes et al., editor, *Proc. Mathematical Foundations of Programming Semantcs (MFPS'93)*, volume 802 of *Lecture Notes in Computer Science*, pages 1–28. Springer, 1993.

[AILS07] L. Aceto, A. Inglfsdttir, K. G. Larsen, and J.. Srba. *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press, 2007.

[Ard60] D. N. Arden. Delayed logic and finite state machines. In *Theory of Computing Machine Design*, pages 1–35. Univ. of. Michigan Press, 1960.

[Bak71] J. W. de Bakker. *Recursive Procedures*. Mathematical Centre Tracts 24, Mathematisch Centrum, Amsterdam, 1971.

[Bak75] J. W. de Bakker. The fixed-point approach in semantics: theory and applications. In J.W. de Bakker, editor, *Foundations of Computer Science*, pages 3–53. Mathematical Centre Tracts 63, Mathematisch Centrum, Amsterdam, 1975.

[BE87] J. Barwise and J. Etchemendy. *The Liar: an Essay in Truth and Circularity*. Oxford University Press, 1987.

[Bek69] H. Bekič. Definable operations in general algebras and the theory of automata and flowcharts. Unpublished Manuscript, IBM Lab. Vienna 1969. Also appeared in [Jon84], 1969.

[Ben76] J. van Benthem. *Modal Correspondence Theory*. PhD thesis, Mathematish Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam, 1976.

[Ben83] J. van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, 1983.

[Ben84] J. van Benthem. Correspondence theory. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 2, pages 167–247. Reidel, 1984.

[Ber54] P. Bernays. A system of axiomatic set theory–Part VII. *J. Symb. Log.*, 19(2):81–96, 1954.

[BGM71] J. Barwise, R. O. Gandy, and Y. N. Moschovakis. The next admissible set. *J. Symb. Log.*, 36:108–120, 1971.

[BGS92] José L. Balcázar, Joaquim Gabarró, and Miklos Santha. Deciding Bisimilarity is P-Complete. *Formal Asp. Comput.*, 4(6A):638–648, 1992.

[BH97] Michael Brandt and Fritz Henglein. Coinductive axiomatization of recursive type equality and subtyping. In Roger Hindley, editor, *Proc. 3rd Conference on Typed Lambda Calculi and Applications (TLCA'97)*, volume 1210 of *Lecture Notes in Computer Science*, pages 63–81. Springer, April 1997.

[Bir48] G. Birkhoff. *Lattice theory (revised edition)*, volume 25 of American Mathematical Society Colloquium Publications. American Mathematical Society, 1948.

[Bli77] Andrzej Blikle. A comparative review of some program verification methods. In Jozef Gruska, editor, *6th Symposium on Mathematical Foundations of Computer Science (MFCS'77)*, volume 53 of *Lecture Notes in Computer Science*, pages 17–33. Springer, 1977.

[BM96] J. Barwise and L. Moss. *Vicious Circles: On the Mathematics of Non-Wellfounded Phenomena*. CSLI (Center for the Study of Language and Information), 1996.

[Bof68] M. Boffa. Les ensembles extraordinaires. *Bulletin de la Société Mathmatique de Belgique*, XX:3–15, 1968.

[Bof69] M. Boffa. Sur la théorie des ensembles sans axiome de fondement. *Bulletin de la Société Mathmatique de Belgique*, XXXI:16–56, 1969.

[Bof72] M. Boffa. Forcing et negation de l'axiome de fondement. *Académie Royale de Belgique, Mémoires de la classe des sciences, 2e série*, XL(7):1–53, 1972.

[Bou50] N. Bourbaki. Sur le théorème de Zorn. *Arch. Math.*., 2:434–437, 1950.

[BR73] J. W. de Bakker and W. P. de Roever. A calculus for recursive program schemes. In M. Nivat, editor, *Proc. IRIA symposium on on Automata, Languages and Programming, Paris, France, July, 1972*, pages 167–196. North-Holland, 1973.

[Bra78] D. Brand. Algebraic simulation between parallel programs. Research Report RC 7206, Yorktown Heights, N.Y., 39 pp., June 1978.

[BRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[Bur75] William H. Burge. Stream processing functions. *IBM Journal of Research and Development*, 19(1):12–25, 1975.

[Cad72] J. M. Cadiou. *Recursive definitions of partial functions and their computations*. PhD thesis, Computer Science Department, Stanford University, 1972.

[CC79] P. Cousot and R. Cousot. Constructive versions of Tarski's fixed point theorems. *Pacific Journal of Mathematics*, 81(1):43–57, 1979.

[Cla77] Edmund M. Clarke. Program invariants as fixed points (preliminary reports). In *FOCS*, pages 18–29. IEEE, 1977. Final version in *Computing*, 21(4):273–294, 1979. Based on Clarke's PhD thesis, "Completeness and Incompleteness Theorems for Hoare-like Axiom Systems, Cornell University, 1976.

[Coq93] Thierry Coquand. Infinite objects in type theory. In Henk Barendregt and Tobias Nipkow, editors, *TYPES*, volume 806 of *Lecture Notes in Computer Science*, pages 62–78. Springer, 1993.

[Dev63] V. Devidé. On monotonous mappings of complete lattices. *Fundamenta Mathematicae*, LIII:147–154, 1963.

[dR77] Willem P. de Roever. On backtracking and greatest fixpoints. In Arto Salomaa and Magnus Steinby, editors, *Fourth Colloquium on Automata, Languages and Programming (ICALP)*, volume 52 of *Lecture Notes in Computer Science*, pages 412–429. Springer, 1977.

[Ehr61] Andrzej Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fundamenta Mathematicae*, 49:129–141, 1961.

[FH83] M. Forti and F. Honsell. Set theory with free construction principles. *Annali Scuola Normale Superiore, Pisa, Serie IV*, X(3):493–522, 1983.

[Fin26] P. Finsler. Über die Grundlagen der Mengenlehre. *I. Math. Zeitschrift*, 25:683–713, 1926.

[Flo67] R. W. Floyd. Assigning meaning to programs. In *Proc. Symposia in Applied Mathematics*, volume 19, pages 19–32. American Mathematical Society, 1967.

[Fra22] A. Fraenkel. Zu den Grundlagen der Cantor-Zermeloschen Mengenlehre. *Math. Annalen*, 86:230–237, 1922.

[Fra53] Roland Fraïssé. *Sur quelques classifications des systmès de relations.* Thesis, University of Paris, 1953. Also in Publications Scientifiques de l'Universit d'Alger, series A 1, 35–182, 1954.

[Fri73] H. Friedman. The consistency of classical set theory relative to a set theory with intuitionistic logic. *J. Symb. Log.*, 38:315–319, 1973.

[Gim96] Eduardo Giménez. *Un Calcul de Constructions Infinies et son Application a la Verification des Systemes Communicants.* PhD thesis PhD 96-11, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, December 1996.

[Gin68] A. Ginzburg. *Algebraic Theory of Automata*. Academic Press, 1968.

[Gla90] R.J. van Glabbeek. The linear time-branching time spectrum (extended abstract). In Jos C. M. Baeten and Jan Willem Klop, editors, *First Conference on Concurrency Theory (CONCUR'90)*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer, 1990.

[Gla93] R.J. van Glabbeek. The linear time — branching time spectrum II (the semantics of sequential systems with silent moves). In E. Best, editor, *Fourth Conference on Concurrency Theory (CONCUR'93)*, volume 715. Springer, 1993.

[GM76] F. Giarratana, V. Gimona and U. Montanari. Observability concepts in abstract data type specification. In A. Mazurkievicz, editor, *5th Symposium on Mathematical Foundations of Computer Science*, volume 45 of *Lecture Notes in Computer Science*, pages 576–587. Springer, 1976.

[Gol89] R. Goldblatt. Varieties of complex algebras. *Ann. Pure Applied Logic*, 44:173–242, 1989.

[Gor82] L. Gordeev. Constructive models for set theory with extensionality. In A.S. Troelstra and D. van Dalen, editors, *The L.E.J. Brouwer Centenary Symposium*, pages 123–147, 1982.

[GR62] Seymour Ginsburg and H. Gordon Rice. Two families of languages related to algol. *J. ACM*, 9(3):350–371, 1962.

[GRS79] John S. Gourlay, William C. Rounds, and Richard Statman. On properties preserved by contraction of concurrent systems. In Gilles Kahn, editor, *International Symposium on Semantics of Concurrent Computation*, volume 70 of *Lecture Notes in Computer Science*, pages 51–65. Springer, 1979.

[HE67] J. van Heijenoort (Ed.). *From Frege to Gödel: A source book in mathematical logic 1879-1931.* Harvard University Press, 1967.

[Hin80] R. Hinnion. Contraction de structures et application à NFU. *Comptes Rendus Acad. des Sciences de Paris*, 290, Sér. A:677–680, 1980.

[Hin81] R. Hinnion. Extensional quotients of structures and applications to the study of the axiom of extensionality. *Bulletin de la Société Mathmatique de Belgique*, XXXIII (Fas. II, Sér. B):173–206, 1981.

[Hin86] R. Hinnion. Extensionality in Zermelo-Fraenkel set theory. *Zeitschr. Math. Logik und Grundlagen Math.*, 32:51–60, 1986.

[HM80] Matthew Hennessy and Robin Milner. On observing nondeterminism and concurrency. In J. W. de Bakker and Jan van Leeuwen, editors, *Proc. 7th Colloquium Automata, Languages and Programming*, volume 85 of *Lecture Notes in Computer Science*, pages 299–309. Springer, 1980.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32:137–161, 1985.

[Hoa72] T. Hoare. Proof of correctness of data representations. *Acta Informatica*, 1:271–281, 1972.

[Hon81] F. Honsell. Modelli della teoria degli insiemi, principi di regolarità e di libera costruzione. Tesi di Laurea, Universita' di Pisa, 1981.

[HP73] P. Hitchcock and D. Park. Induction rules and termination proofs. In M. Nivat, editor, *Proc. IRIA symposium on on Automata, Languages and Programming, Paris, France, July, 1972*, pages 225–251. North-Holland, 1973.

[Huf54] D.A. Huffman. The synthesis of sequential switching circuits. *Journal of the Franklin Institute (Mar. 1954) and (Apr. 1954)*, 257(3–4):161–190 and 275–303, 1954.

[Imm82] Neil Immerman. Upper and lower bounds for first order expressibility. *J. Comput. Syst. Sci.*, 25(1):76–98, 1982.

[Jen80] Kurt Jensen. A method to compare the descriptive power of different types of petri nets. In Piotr Dembinski, editor, *Proc. 9th Mathematical Foundations of Computer Science 1980 (MFCS'80), Rydzyna, Poland, September 1980*, volume 88 of *Lecture Notes in Computer Science*, pages 348–361. Springer, 1980.

[Jon84] Cliff B. Jones, editor. *Programming Languages and Their Definition – Hans Bekic (1936-1982)*, volume 177 of *Lecture Notes in Computer Science*. Springer, 1984.

[JR96] Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the EATCS*, 62:222–259, 1996.

[JT66] D.H.J. de Jongh and A.S. Troelstra. On the connection of partially ordered sets with some pseudo-boolean algebras. *Indagationes Mathematicae*, 28:317–329, 1966.

[Kah74] Gilles Kahn. The semantics of simple language for parallel programming. In *IFIP Congress*, pages 471–475, 1974.

[Kan39] L. V. Kantorovich. The method of successive approximations for functional equations. *Acta Math.*, 71:63–97, 1939.

[Kle52] S. C. Kleene. *Introduction to Metamathematics*. Van Nostrand, 1952.

[Kle70] S. C. Kleene. The origin of recursive function theory. In *20th annual Symposium on Foundations of Computer Science*, pages 371–382. IEEE, 1970.

[Kna28] B. Knaster. Un théorèm sur les fonctions d'ensembles. *Annals Soc. Pol. Math*, 6:133–134, 1928.

[KS90] Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Inf. Comput.*, 86(1):43–68, 1990.

[Kwo77] Y. S. Kwong. On reduction of asynchronous systems. *Theoretical Computer Science*, 5(1):25–50, 1977.

[Lan64] Peter J. Landin. The mechanical evaluation of expressions. *The Computer Journal*, 6(4):308–320, 1964.

[Lan65a] Peter J. Landin. Correspondence between ALGOL 60 and Church's Lambda-notation: Part I. *Commun. ACM*, 8(2):89–101, 1965.

[Lan65b] Peter J. Landin. A correspondence between ALGOL 60 and Church's Lambda-notations: Part II. *Commun. ACM*, 8(3):158–167, 1965.

[Lan69] P. Landin. A program-machine symmetric automata theory. *Machine Intelligence*, 5:99–120, 1969.

[LNS82] J.-L. Lassez, V. L. Nguyen, and L. Sonenberg. Fixed point theorems and semantics: A folk tale. *Inf. Process. Lett.*, 14(3):112–116, 1982.

[Man69] Z. Manna. The correctness of programs. *J. Computer and System Sciences*, 3(2):119–127, 1969.

[Man74] Z. Manna. *Mathematical Theory of Computation*. McGraw-Hill, 1974.

[Maz71] Antoni W. Mazurkiewicz. Proving algorithms by tail functions. *Information and Control*, 18(3):220–226, 1971.

[Maz73] A. Mazurkiewicz. Proving properties of processes. Tech. rep. 134, Computation Center ol Polish Academy of Sciences, Warsaw, 1973. Also in Algorytmy 11, 5–22, 1974.

[McC61] J. McCarthy. A basis for a mathematical theory of computation. In *Proc. Western Joint Computer Conf.*, volume 19, pages 225–238. Spartan Books, 1961. Reprinted, with corrections and an added tenth section, in Computer Programming and Formal Systems, P. Braffort and D. Hirschberg, eds., North-Holland, 1963, pp. 33–70.

[McC63] J. McCarthy. Towards a mathematical science of computation. In *Proceedings of IFIP Congress 62*, pages 21–28. North-Holland, 1963.

[Mil70] R. Milner. A formal notion of simulation between programs. Memo 14, Computers and Logic Resarch Group, University College of Swansea, U.K., 1970.

[Mil71a] R. Milner. Program simulation: an extended formal notion. Memo 17, Computers and Logic Resarch Group, University College of Swansea, U.K., 1971.

[Mil71b] R. Milner. An algebraic definition of simulation between programs. In *Proc. 2nd Int. Joint Conferences on Artificial Intelligence*. British Comp. Soc. London, 1971.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[Mir17a] D. Mirimanoff. Les antinomies de Russell et de Burali-Forti et le problème fondamental de la théorie des ensembles. *L'Enseignement Mathématique*, 19:37–52, 1917.

[Mir17b] D. Mirimanoff. Remarques sur la théorie des ensembles et les antinomies cantoriennes I. *L'Enseignement Mathématique*, 19:209–217, 1917.

[Mir20] D. Mirimanoff. Remarques sur la théorie des ensembles et les antinomies cantoriennes II. *L'Enseignement Mathématique*, 21:29–52, 1920.

[Moo56] E.F. Moore. Gedanken experiments on sequential machines. *Automata Studies, Annals of Mathematics Series*, 34:129–153, 1956.

[Mor68] James H. Morris. *Lambda-Calculus Models of Programming Languages*. Phd thesis MAC-TR-57, M.I.T., project MAC, Dec. 1968.

[Mos74] Yiannis Nicholas Moschovakis. *Elementary induction on abstract structures*, volume 77 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1974.

[MS72] Albert R. Meyer and Larry J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *13th Annual Symposium on Switching and Automata Theory (FOCS*, pages 125–129, 1972.

[MT91] R. Milner and M. Tofte. Co-induction in relational semantics. *Theoretical Computer Science*, 87:209–220, 1991. Also Tech. Rep. ECS-LFCS-88-65, University of Edinburgh, 1988.

[Ner58] A. Nerode. Linear automaton transformations. In *Proc. American Mathematical Society*, volume 9, pages 541–544, 1958.

[Par69] D. Park. Fixpoint induction and proofs of program properties. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 59–78. Edinburgh Univ. Press, 1969.

[Par70] D. Park. The Y-combinator in Scott's lambda-calculus models. Symposium on Theory of Programming, University of Warwick, unpublished (A revised version: Research Report CS-RR-013, Department of Computer Science, University of Warwick, June 1976.), 1970.

[Par79] D. Park. On the semantics of fair parallelism. In *Proc. Abstract Software Specifications, Copenhagen Winter School*, Lecture Notes in Computer Science, pages 504–526. Springer, 1979.

[Par81a] D. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conf. on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[Par81b] D. Park. A new equivalence notion for communicating systems. In G. Maurer, editor, *Bulletin EATCS*, volume 14, pages 78–80, 1981. Abstract of the talk presented at the Second Workshop on the Semantics of Programming Languages, Bad Honnef, March 16–20 1981. Abstracts collected in the Bulletin by B. Mayoh.

[Pas74] Antonio Pasini. Some fixed point theorems of the mappings of partially ordered sets. *Rendiconti del Seminario Matematico della Università di Padova*, 51:167–177, 1974.

[Pou07] Damien Pous. Complete lattices and up-to techniques. In *5th Asian Symposium on Programming Languages and Systems (APLAS)*, volume 4807 of *Lecture Notes in Computer Science*, pages 351–366. Springer, 2007.

[PT87] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[Rey93] John C. Reynolds. The discoveries of continuations. *Lisp and Symbolic Computation*, 6(3-4):233–248, 1993.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, 1967. Reprinted, MIT Press 1987.

[RT92] J. Rutten and D. Turi. On the foundation of final semantics: Non-standard sets, metric spaces, partial orders. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX Workshop*, volume 666 of *Lecture Notes in Computer Science*, pages 477–530. Springer, 1992.

[Rus03] B. Russell. *Principles of Mathematics*. Cambridge University Press, 1903.

[Rus08] B. Russell. Mathematical logic as based on the theory of types. *American J. of Mathematics*, 30:222–262, 1908. Also in [HE67], pages 153–168.

[RW13] B. Russell and A. N. Whitehead. *Principia Mathematica, 3 vols.* Cambridge University Press, 1910, 1912, 1913.

[San98] D. Sangiorgi. On the bisimulation proof method. *Journal of Mathematical Structures in Computer Science*, 8:447–479, 1998.

[Sco60] D. Scott. A different kind of model for set theory. Unpublished paper, given at the 1960 Stanford Congress of Logic, Methodology and Philosophy of Science, 1960.

[Sco69a] D. Scott. Models for the λ-calculus. Manuscript, raft, Oxford, December 1969.

[Sco69b] D. Scott. A construction of a model for the λ-calculus. Manuscript, Oxford, November 1969.

[Sco69c] D. Scott. A type-theoretical alternative to CUCH, ISWIM, OWHY. Typed script, Oxford. Also appeared as [Sco93], October 1969.

[Sco72a] D. Scott. Continuous lattices. In E. Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. Springer, 1972.

[Sco72b] D. Scott. The lattice of flow diagrams. In E. Engeler, editor, *Symposium of Semantics of Algorithmic Languages*, volume 188 of *Lecture Notes in Mathematics*, pages 311–366. Springer, 1972.

[Sco76] D. Scott. Data types as lattices. *SIAM J. on Computing*, 5:522–587, 1976. An manuscript with the same title was written in 1972.

[Sco93] Dana S. Scott. A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121(1&2):411–440, 1993.

[SdB69] D. Scott and J.W. de Bakker. A theory of programs. Handwritten notes. IBM Lab., Vienna, Austria, 1969.

[Seg68] K. Segerberg. Decidability of S4.1. *Theoria*, 34:7–20, 1968.

[Seg70] K. Segerberg. Modal logics with linear alternative relations. *Theoria*, 36:301–322, 1970.

[Seg71] Krister Segerberg. An essay in classical modal logic. Filosofiska Studier, Uppsala, 1971.

[Sko23] T. Skolem. Einige Bemerkungen zur axiomatischen Begründung der Mengenlehre. In Helsinki Akademiska Bokhandeln, editor, *Proceedings of the 5th Scandinavian Mathematics Congress, Helsinki, 1922*, pages 217–232, 1923. English translation, "Some remarks on axiomatized set theory", in [HE67], pages 290–301.

[SKS07] Davide Sangiorgi, Naoki Kobayashi, and Eijiro Sumii. Environmental bisimulations for higher-order languages. In *22nd Annual IEEE Symposium on Logic in Computer Science (LICS'07)*, pages 293–302. IEEE Computer Society, 2007.

[Spe57] E. Specker. Zur Axiomatik der Mengenlehre. *Z. Math. Logik*, 3(3), 1957.

[SW01] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[Tar49] Alfred Tarski. A fixpoint theorem for lattices and its applications (preliminary report). *Bull. Amer. Math. Soc.*, 55:1051–1052 and 1192, 1949.

[Tar55] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pac. J. Math.*, 5:285–309, 1955.

[Tho72] S. K. Thomason. Semantic analysis of tense logics. *J. Symb. Log.*, 37(1):150–158, 1972.

[Tho93] Wolfgang Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science. In Marie-Claude Gaudel and Jean-Pierre Jouannaud, editors, *TAPSOFT*, volume 668 of *Lecture Notes in Computer Science*, pages 559–568. Springer, 1993.

[TP97] Daniele Turi and Gordon D. Plotkin. Towards a mathematical operational semantics. In *12th Annual IEEE Symposium on Logic in Computer Science (LICS'97)*, pages 280–291. IEEE Computer Society Press, 1997.

[Zer08] E. Zermelo. Untersuchungen über die Grundlagen der Mengenlehre I. *Mathematische Annalen*, 65:261–281, 1908. English translation, "Investigations in the foundations of set theory", in [HE67], pages 199-215.